A REVIEW OF THREE EDITING AND IMPUTATION SYSTEMS

Mark Pierzchala, National Agricultural Statistics Service (NASS)
Room 4168 South Building, USDA, Washington, DC 20250-2000, 202-447-5333

KEY WORDS: Error Localization, Edit Analysis, Expert System, CATI, CAPI, Survey Integration, GEIS, Blaise, SPEER, Fellegi and Holt systems.

Systems Evaluated

Blaise [1], version 2.22, Netherlands Central Bureau of Statistics
Generalized Edit and Imputation System (GEIS), version 6.1, Statistics Canada
Structured Program for Economic Editing and Referrals (SPEER), U. S. Bureau of the Census

Checklist

The three systems were evaluated in part by using a checklist of about 250 editing system features. The checklist was written by the author but is based heavily on the work of Paul Cotton (1988). Versions have been reviewed extensively by many people. A copy can be found in the report of the Subcommittee on Data Editing in Federal Statistical Agencies (1990). A copy of the blank checklist in either ASCII or Word Perfect format may be obtained from the author.

Method of Evaluation

Nine questionnaires, including NASS's biggest and most complicated, have been programmed into Blaise by various people in the agency. By contrast, one section each from two questionnaires have been programmed into GEIS, and only one section from one questionnaire has been programmed into SPEER. This latter section was programmed into all three systems for the purposes of this evaluation. On the other hand, references to other sources of information has been fairly even as demonstrated in Table 1.

Table 1. Author's Familiarity with the Systems

Evaluation Method	Blaise	GEIS	SPEER
Programming (notes)	9 forms (2 as CAPI)	65 vars (fr 2 forms)	45 vars (fr 1 form)
Papers read (about)	6	10	8
Documentation	Yes	Yes	None
Discussions	Many	Many	Many
Demonstrations by team members	2	2	3
Checklist	Yes	Yes	Yes

Processing Flow

Before it is possible to discuss the three systems, it is necessary to describe which functions of the survey process each system addresses. Following is a useful table.

Table 2. Parts of Processing Flow Handled

Function	Blaise	GEIS	SPEER
Pre-survey edit analysis (FH)		**Yes**	**Yes**
Primary data entry after collection (I)	Yes		
CATI / CAPI (I) [2]	Yes		
Complex coding (I)	Yes		
On-line, manual edit	Yes		Yes
Automated error localization (FH)		**Yes**	**Yes**
Model-based imputation	Yes	Yes	Yes
Hot- / cold-deck imputation		Yes	
Macro-editing	~	~	~
Survey management reports	Yes	Yes	Yes

Functions that are specific to Fellegi and Holt systems are bolded and labelled (FH). Those that are specific to integration systems are underlined and labelled (I). Blaise is an integration system, while GEIS and SPEER are Fellegi and Holt systems.

Integration Contrasted with Fellegi and Holt

An integration system seeks to perform as many different survey functions as possible within one system. Common to all integration systems is a "specifications generator". In a specifications generator, all information about the questionnaire is stated, such as data ranges, routes, file structure, and edits. This information is used to generate separate modules for data collection, data entry, editing, and the like. Time is saved by the elimination of multiple specification of the data and by eliminating conversions from one data set to another. Where a person must perform a function, such as questionnaire review or coding, it is performed on-line, eliminating cyclic pro-

cessing. In addition, data file setups for databases and statistical packages may be automatically generated from the information stated in the specifications generator.

By contrast, Fellegi and Holt systems are more focused on the post-collection micro-editing part of the survey process. They process data in batch though SPEER has an on-line aspect also. The role of the data editor is reduced in the processing of data and increased in the pre-survey analysis of edit specifications. Though Fellegi and Holt systems reduce the amount of editing done by people, there is still a need for a considerable amount of pre-processing. Imputation can be almost totally automated, and can be either model-based, or donor-based. Fellegi and Holt (1976) state four principles of editing: 1) that each record should satisfy all edits, 2) that correction should be accomplished by making as few changes as possible, 3) that editing and imputation are part of the same process, and 4) that the imputation process should retain the structure of the data.

Fellegi and Holt also give a mathematical procedure for edit and imputation of categorical data. In this procedure, explicitly written edits are used to generate implicit edits. For example (to take a case in the continuous domain) if $1 < a/b < 2$ and $3 < b/c < 4$ are explicit edits, then $3 < a/c < 8$ is an implied edit. These implied edits are used in two ways. First they are inspected to determine if the data are being constrained in an undetermined way. Second, the implied edits taken together with the explicit edits form a "feasible region". According to Fellegi and Holt, if imputations are made such that the imputations fall within the feasible region, then all the edits are automatically satisfied. Error localization is the process where the system determines for which fields imputations should be made. GEIS and SPEER extend the theory of Fellegi and Holt into the continuous realm.

General Features

Before an organization should undertake a formal evaluation of an editing system, four questions must be asked: 1) Which kinds of data and surveys must be handled? 2) Where must it fit into the survey processing flow? 3) Which environments must it operate in? 4) Which kinds of edits can it handle? For the three systems in this paper, the answers to question 2 has already been provided in Table 2. The answers to questions 1, 3 and 4 appear below in Tables 3, 4, and 5.

Table 3. Kinds of Surveys and Data Handled

Survey Type	Blaise	GEIS	SPEER
Economic	Y	Y	Y
Social	Y		
Demographic	Y		
Data Type			
Continuous	Y	Y	Y
Categorical	Y		
Character	Y		

Table 4. Environments and Subsidiary Software

Operating System	Blaise	GEIS	SPEER
MS DOS	Y	Y	Y
OS / 2	~	~	~
UNIX		Y	
MVS		Y	
CMS	~	~	~
VMS			Y
Hardware			
Personal Computer	Y	Y	Y
Local Area Network	Y		Y
Mainframe / mini		Y	Y
Subsidiary Software			
Appl Developer	Turbo Pascal	Oracle	Fortran
End User		Oracle	
Mode of Use			
Batch	Y	Only	Y
On-line	Primary		Y

Table 5. Types of Edits Handled

Type of Edit	Blaise	GEIS	SPEER
Linear	Y	Y	
Ratio	Y	Y#	Y
Conditional	Y		
Existency	Y	Y#	
Consistency	Y	Y	Y#
Additivity	Y	Y	Y%
Multiplicativity	Y	Y#	Y#
Bounds	Y	Y	Y%
Route	Y		
Valid Value	Y		
Historical	Y		
Levels of failure	Y		continuum
Stat Outlier Detection		Y%	

Y#, Requires some kind of restatement; Y%, Not part of edit analysis; Y, Primary edit type.

GEIS and SPEER are limited to certain kinds of edits and to continuous data. This is due to the mathematics involved in edit analysis and error localization.

Blaise

Blaise Vision

The functions of data entry, editing, computer assisted data collection, coding, and survey management are integrated in one system. Instruments are programmed by subject matter specialists while systems development is handled by systems personnel. Development and processing are done in a paperless environment. The system must be easy to use and, where appropriate, menu driven. The data editor is held to be fully knowledgeable and competent to make all necessary changes. The system must be able to handle all data types, questionnaire sizes, and data structures encountered in the Netherlands Central Bureau of Statistics (CBS). Finally the system is to operate either on LANs (for in-house work) or on portable computers (for CAPI). It is the standard editing system of the bureau.

How Productivity is Promoted

Since the whole bureau uses a standard system the subject matter departments are relieved from the necessity of having to produce, test, and maintain their own systems. By using an integration system with its data dictionary, one programmer can develop an application in all of its manifestations. For example, one instrument can be written for CATI and editing. By entering the data into and editing within one instrument, and then having the system produce a file setup for a statistical package such as SPSS, much time is saved by eliminating and reducing conversions from one system to another. The interactive editing environment promotes productivity by eliminating batch processing on printouts. In the batch environment, it may take several days to clean a form. In the interactive environment, the results of changes are immediately known, thus the questionnaire is handled once. As the editor enters data or changes directly, the need for data entry is reduced. Where one system is standard, expertise is built up throughout the bureau facilitating exchange of people and ideas between units.

How Quality is Improved

Quality of the data is improved through the promotion of computer assisted data collection (CATI, CAPI, and self-administered questionnaires). Here editing is included in the data collection where the respondent can help clear up problems. As a result, post-collection editing is reduced. Any differences between collection and editing instruments are purposeful, not accidental.

For surveys that are collected on paper, quality of the process is improved because each form can be cleaned in one session. The editor does not have to try to recall what she was doing with the form one or more days ago. Also, as the editor works with the instrument she learns which mistakes are made most often by the interviewers and can give them feedback. This is because all errors must be corrected or the error signals must be suppressed. The file of suppressed error signals can be used to monitor editor actions. As data are processed more quickly, the timeliness quality of the data is improved.

Types of Edits

Virtually any kind of mathematical statement can be used as an edit in the Blaise system as shown in Table 5. As Blaise is not a Fellegi and Holt system, it is not constrained by the underlying mathematics to a specific data type (i.e., continuous vs. categorical), nor to types of edits (e.g., linear mathematical expressions). Extremely complicated conditional edits of great length are allowed. Edits based on external data can be written, either by reading other Blaise data files or ASCII files. Some edits are implicitly stated either in data definition or in route statements (as in a CAPI instrument). Model based statistical edits are possible if the proper information is made available through external files. Two levels of edit failure are allowed, suspicious and dirty. The suspicious error signals can be suppressed, the dirty ones cannot.

Survey Management

Survey management here is defined as generating information on the progress of the survey and being able to act on that information by making forms go where they need to go. Blaise has some survey management capabilities but these can be improved.

Abacus is a tabling program that can be used to produce reports on response rates, processing flows, cleanliness status etc., for each application. Tables are interactively generated. Abacus operates either on Blaise or ASCII files. It can produce counts, totals, averages, and percentages though it is not meant to be a full-fledged summary system. Continuous variables can be summarized in categories. The Blaise system generates an Abacus setup so that data definition does not have to be repeated. This latter feature has broken down on occasion for large questionnaires (it is being fixed in version 2.3). As a result the setups had to be corrected once or twice. First attempts to produce an Abacus table are likely to be confusing and the booklet that comes with Abacus does not contain enough examples. However, the online documentation is context-sensitive and good, and once learned, the programming is easy and fast. Table specifications can be saved and called from DOS so that the end users do not have to generate them from menus.

Blaise has fairly good flexibility in the choice of records to be read out of the system, based on the status Clean, Suspicious, or Dirty, and on whether the report is New (never been read out), Old (been previously read out), or Changed (been previously read out but has been subsequently changed). More flexibility is needed including the ability to read out records by ID or by class (e.g., stratum number). The same flexibility is needed in the batch check as this tends to be an all or nothing affair. The editor has great flexibility in choosing records to call up for interactive processing including access by ID, values of designated variables, and status of the report.The Blaise indexed file structure allows for fast access to the chosen record.

Blaise does not have an audit trail nor does it automatically generate error counts. The lack of these features can probably be attributed to the vision of how the CBS editor is to do her job. Typically she will enter and edit the data at the same time. Hence data will in some cases be changed as they are entered (before they are captured). An audit trail and error counts do not have as much value in this scenario because of the many changes that are made before they can be generated. In one survey, NASS has used Abacus to generate error counts but this involved programming indicator variables in Blaise to indicate when an error was violated. This is not a very satisfactory solution because if an edit statement must be changed, the programmer must remember to change the calculation for the indicator variable as well. As well, NASS has captured raw data in Blaise and edited a copy of the raw data file. Then the two files were compared in SAS. This is an advance from current procedures where an intensive hand edit is done before data capture, but it is not enough. These gaps are methodological in nature. Properly summarized audit trails and error counts can be used to provide feedback on the performance of the survey, assuming that data are captured before the edit.

System Items

Blaise has been developed in modules which facilitates further system improvement. For the applications developer, an integrated development environment is provided including a powerful easy-to-use text editor, a syntax checker, and a compiler. If the syntax checker detects an error, checking will stop, a message will be displayed, and the programmer will be asked if he wants to invoke the text editor. If yes, the system will place the cursor at the spot where the error was detected. This is the correct place about 98% of the time. On the other hand, the syntax checker will catch only one error at a time and must start at the beginning of the instrument each time it is invoked. Some developers in NASS have commented that they would like the syntax checker to catch several errors at once. Utilities for defining external files or for reading data in and out of Blaise are part of the system and are accessed through the menus. Automated file setups for SPSS, Paradox, Stata, and Abacus are provided as they are standard packages in the CBS. In order to connect with other packages, a wide variety of ASCII setups can be generated. The Blaise system is accessed with an easy-to-use and dynamic menu system. Some facilities such as reading files in and out of Blaise can be done either through batch calls or the menus. The coding facility allows either step-wise or dictionary coding and has worked well in one small NASS survey.

Features that should be added are the ability to make system calls for user defined functions and the capability to generate file setups for additional packages such as SAS.

Edit Writing (Instrument Writing)

Instruments are written in Blaise, a structured language. There is a paragraph each for data definition, route statements, suspicious errors, and dirty errors. This seems to facilitate programming for the subject matter specialist as it helps him to keep organized. For complicated instruments, the paragraphs can be built into blocks and the instrument built from the blocks. Once a block is built, it can be used many times, in the same instrument or in different instruments. Tables up to thousands of cells are fairly easy to set up in Blaise and have powerful repetitive capabilities. Boolean operators (AND, OR, ELSE, IN, NOT) greatly facilitate programming.

Feasible regions are defined in Blaise edits. A typical edit in Blaise might look like:

```
If stratum <= 50 then
 20 <= farmsize <= 200 "The farm size
 is usually between 20 and 200
 acres.";
endif;
```

External files can be referenced in edits, but are somewhat difficult to specify. There are no GOTO statements in Blaise, a fact that the CBS heralds as a great advance in instrument design. This feature, an integral part of the structured approach, certainly makes it much easier to determine what is happening in the code when debugging than with systems that use GOTO statements.

Blaise instruments are meant to be written by subject matter specialists. In NASS, this has proved to be possible as several small to medium instruments have been written by state office personnel with little or no training and some support from the author. In headquarters, large instruments have been programmed, again without training and very little support from the CBS. On the other hand, NASS does not use the full power of Blaise and thus some instruments are longer and more complicated than necessary.

Data Review and Correction

This feature is one of the stronger points of editing in Blaise. Data review and correction occur primarily through on-line sessions although deletions and imputations can be carried out in batch as well. Blaise has a very dynamic user interface with menus for form selection, pop-up windows for messages, full-screen mobility using Pg Up, Pg Down, arrows, and some function keys. Especially valuable are the Ctl End and Ctl Home keys which get the editor to the next cell involved in an error and pops up the error message. This allows errors to be found very quickly in large instruments which take up many screens. Variables must all be named (e.g., farmsize) but can be numbered as well. The numbers can be used to jump from one variable to another which is another quick way of navigating through a large questionnaire. Errors are signalled by numbers which appear to the left of the cell value. Suspicious (soft) error signals are flagged by unbolded numbers while dirty (hard) error signals are flagged by bolded numbers. The number that makes up the error signal tells how many edit failures the cell value is involved in. For example, the numbers '12' (read one, two), would indicate that the cell value is involved in one suspicious edit failure and two dirty edit failures. The editor can re-edit the form at any time by pressing the F3 key thus gaining immediate feedback on his actions. Suspicious errors signals can be suppressed, dirty ones cannot (though in NASS a procedure has been invented that allows selected dirty signals to be suppressed). Blaise tables allow the editor to view a large number of variables at one time.

A few improvements can be made. For example, a counter should be displayed at the top of the screen that tells the editor how many forms are left to be inspected. An "undo" option for the last editor action would also be a nice addition (currently, all changes can be undone).

Support, Updates, and Training

Written documentation is available in English and is quite good as far as it goes. Users can program sophisticated applications by reference to the manuals and without training though in this case the full power of the system is not likely to be exploited. Documentation consists of an Introduction, a Language Reference Manual, a System Description, a manual for editors, a manual for interviewers (for CATI and CAPI), and a short manual for the Abacus tabulation package. Needed further is an applications manual which would guide the applications programmer from start to finish in producing the instrument. Updates come every six months to a year. New versions come every year or two. Each new update or version has had some nice enhancements.

Blaise is sold "as is" meaning that no support is offered. This fact puts off a lot of potential buyers. The CBS may yet offer support for the system. Why, one must ask, does the CBS produce the system in four languages (Dutch, English, French, and Spanish), provide a method for conversion to other languages, and documentation in English as well as in Dutch? It would be nice if support were offered either directly or through a North American contact and a training course held in North America at least once a year. This could easily be accomplished as the CBS regularly conducts training for its own employees.

The current version of Blaise is 2.3 which includes a forms manager and call scheduler for CATI.

Integration

NASS has done some initial evaluation of Blaise as a CAPI system. Two large instruments have been created, one of which was tested in the field and worked well. The other, representing NASS's largest survey, will be field tested in early 1991. This latter instrument will include about 1000 variables. After the CAPI instrument is finished, it will be converted to an editing instrument. The CBS claims that an editing instrument and a CAPI/CATI instrument can be created from the same code by compiling the code in different modes. This is true, however agency policy may prohibit some edits from being used in data collection. In any case, it is better to start with the CAPI instrument and then convert it to an editing instrument than vice versa because the enumerator must use the instrument in a much more sensitive environment. For example, question wording must be exactly correct for data collection, it does not have to exist at all for editing.

NASS has also used Blaise for high-speed data entry in about three surveys, including item-code data entry in one survey. Data entry has worked well. It does not have all of the features of a commercial data entry program. For example, verification is not possible and decimal points must be keyed. In surveys where verification is required, NASS has used Key Entry III for data entry and then has read the data into Blaise for editing.

Methodological Comments and Summary

Overall, Blaise is a very good system that is undergoing continual enhancement. The author and an Interactive Editing Working Group in NASS have recommended its purchase for data editing in the agency. Primary concerns are methodological in that Blaise lacks an audit trail or some other way of keeping track of what changes are being made to the form. On the other hand, Blaise works to reduce the number of errors made in the first place through its promotion of computer assisted data collection and integration of survey functions.

GEIS

GEIS Vision

Applications development is carried out by a subject matter specialist in conjunction with a methodologist. The development is carried out through menus and by entering information onto ORACLE screens. Edit analysis tools are used before data are collected in order to fine-tune the edits. Development can be carried out on a microcomputer using the menu system, however major testing and actual production are run in batch mode on mainframes. Methodologists and systems people work to develop the system. The functioning of the whole system is organized through the database system ORACLE for which the knowledge of a few procedures and commands is sufficient. GEIS is meant to handle all economic collections in Statistics Canada. It is also just one part of a major redesign project which will also include a Data Capture and Collection (DC2) system, and a generalized summary system.

A preliminary edit is carried out before the application of the GEIS system. All coding and callbacks will be handled in the pre-edit as well as reports of major impact. At some point, however, the data will be given over to GEIS for both methodological and productivity reasons. The subject matter specialist is relieved of the tedium of inspecting thousands of forms while her expertise is expressed in GEIS through edit specification, weighting of the relative reliability of the variables, and by influencing how imputation is done. Data processed in GEIS are not further treated except for the few cases that GEIS cannot handle. As GEIS is a Fellegi and Holt system, the four principles previously stated must be considered part of the GEIS vision.

How Productivity is Promoted

In Canada, large surveys may bring in hundreds of thousands of returns. The preliminary edit may have to deal with many of the problems that require callbacks or special handling, but remaining changes will have minor impact on the final estimates. These can be better handled in GEIS. The system will delete and impute for values according to the Fellegi and Holt methodology, and do it quickly and consistently, saving many person-years of hand processing. The implementation of a tested, standardized system that will operate on a large number of surveys will decrease the time spent in system development, testing, and maintenance by subject matter departments.

How Quality is Improved

GEIS ensures that changes in data are made to as few fields as possible and are made so that edits are satisfied. This way the maximum amount of original data are kept subject to the stipulation that edits are satisfied. Computer actions do not depend upon enumerator judgement and thus are more consistent (i.e., repeatable, reproducible). Imputations are made with the intent of preserving the multivariate structure of the data. To the extent that data are processed more quickly, the timeliness quality of the data is improved. Full reports are generated on the changes made by the system allowing a determination of the impact of the system on the estimates.

Types of Edits

The linear mathematical expression is the basic edit in GEIS. Any edit that can be stated in linear form can be handled by the system, with some caveats. For example, the ratio edit $2 < a/b < 4$ can be expressed as two linear edits $2b < a$ and $a < 4b$. Tricks are applied to accommodate some kinds of edits, for example, if $a > 0$ then $b > 0$. This is done by stating that $b > k * a$ where k is a very small constant. Thus if $a = 0$, then **b can equal 0** also, but if $a > 0$, then $b > 0$ is required. It is apparently not possible in GEIS to *require* that if $a = 0$ then **b must equal 0**. This is a kind of edit that would be better left to the pre-edit. IF-THEN (conditional) expressions are handled in two ways. First is to rewrite the edit so that the IF-THEN is no longer needed. Second is to split the file so that subsets of records are operated on by different sets of edits. Multiplicativity edits are troublesome. It is possible to linearize such an edit with the use of logs but then other edits which involve those same variables must also have logs applied, a process which becomes very confusing. However, 95% or more of all edits that were programmed in GEIS were either linear or ratio to start with or easily linearized. There is also a univariate and bivariate statistical outlier procedure which can also produce edit limits. If the statistical edits are produced from a file of previously collected data, then they can be included in the edit analysis.

The biggest problem is with the number of edits that can be stated at one time. Depending on the complexity of the edits, how they interconnect, and thus how many implied edits are generated, as few as 10 explicit edits can exceed system limits, at least on a microcomputer. Some applications have managed 40 or so explicit edits at once. There is a similar limitation with the number of variables that can be handled at one time with about 30 variables being the usual maximum though this too has been exceeded on occasion.

There is only one degree of edit failure. Designations such as hard or soft errors are foreign to the Fellegi and Hold philosophy.

Edit Writing and Edit Analysis

The writing of edits is done in an ORACLE screen

designed for GEIS input. Either valid or invalid regions can be specified with the valid choice being the preferred statement. Edits must be labeled, to take an extremely small example:

```
ED01 | pass | farmsize <= 200
ED02 | pass | farmsize >= 20
ED03 | pass | farmsize <= 1000
ED04 | pass | farmsize >= 100
ED05 | pass | cropland <= farmsize
```

Edits are then grouped according to how they will be applied to the data. For example:

```
G1 = (ED01, ED02, ED05)
G2 = (ED03, ED04, ED05)
```

Then each group undergoes edit analysis individually and would be applied to different sets of data. For example, group G1 might be applied to stratum 50 and group G2 to stratum 51 where the file has been split.

The writing of the edits is quite easy. Harder is the determination of edit limits, especially for those who are used to writing edits for manual review. In this latter case, limits are often set quite tightly, with the subject matter specialist determining whether a change should be made. In GEIS, since violated edits will cause at least one value to be changed more care must be taken when setting edit limits. To help the specialist set bounds, edit analysis tools are offered. First, implied edits are generated. These have been explained above. For group G1, one implied edit generated from edits ED01 and ED05 is:

```
cropland <= 200
```

This implied edit is obvious but some are not and it is a good way to see if the data are being constrained in an unintended way. A complementary analysis is to determine the worst possible records that can pass the system without invoking an edit failure. This is done with the extremal points option. Extremal points are artificially generated records. For group G1, one extremal point is:

```
cropland = 0
farmsize = 200
```

These two edit analysis procedures are valuable, however for large numbers of variables and edits, the lists of implied edits and extremal points becomes quite lengthy and sometimes cannot even be generated. Additivity edits involving several variables (e.g, a + b + c + d + e = f) greatly increase the numbers of implied edits and extremal points. For example, in one analysis of 17 variables and 11 edits, 3 of which were additivity edits, 384 extremal points were generated and after 3 hours of chugging, no implied edits were ever output. When two of the additivity edits were removed, 76 extremal points and 104 implied edits were generated. These were generated on an MS DOS microcomputer with a 386 processor. I do not know what the result would have been had the same analyses been run on a mainframe.

A much smaller problem with the edit analysis is that the output is hard to read. The implied edits are output in a canonical form such as:

```
- valbuild + valother + valtnt <= 0
 valother - valreal + valtnt <= 0
```

They would be easier to read if they appeared as:

```
valother + valtnt <= valbuild
valother + valtnt <= valreal
```

The extremal points are output in alphabetical order, not in the same order that they appear on the questionnaire. This makes them harder to interpret. To take a fictitious example:

Order on questionnaire	Appearance on printout
acres	**acres = 100**
valland	**assets = 0**
invtot	**invtot = 0**
assets	**valfarm = 200000**
valfarm	**valland = 200000**

Another way to evaluate the edits is to apply them to data. This can be done on old data or on the first N records of the current survey. This method of edit analysis is much used. In other words, the specification of edits tends to be an iterative process.

After imputation, follow-up edits are applied to make sure that the imputations have satisfied the edits. These follow-up edits are similar to the first set of edits except that the bounds are slightly broader.

A further problem in writing edits is due to systems limitations. Large questionnaires must be broken into several parts. The difficulty is in handling variables that are involved in edits in two or more sections. Suppose variable x is found in edits in sections A and B and A is processed before B. If the value of x is changed in A then that change can be taken into account in B. However, if the value of x is changed in B, then there is no way for that change to be taken into account in A. This is handled by giving these variables a high weight in the second section which reduces their possibility of being changed.

Survey Management

In GEIS, where the system takes all actions, the only way that the statistician knows what is happening is through reports generated by the system. GEIS provides an abundance of automatically generated reports. The reporting capability of a database system such as ORACLE and the existence of an audit trail give GEIS extraordinary capability of reporting on the effects of editing ac-

tions. Most tables are easy to read though some of them employ jargon that is not defined in the documentation. Reports are provided on edit analyses, results of applications of edits, fields chosen for deletion, records chosen for matching, results of imputations, plus many more.

In this evaluation, data were not run on a mainframe. In speaking with the GEIS team, the microcomputer may be used to set up specifications for mainframe processing. On the mainframe these specifications are linked together with systems commands. Apparently some "hand channeling" of jobs is still necessary though this will diminish with time.

Systems Items

GEIS is constructed in modules to facilitate development. It is portable across architectures and operating systems. The computationally intensive routines are written in C which speeds processing. Development of applications is done through menus and well-defined screens. The ORACLE database environment provides great flexibility in the reading out of records and for data extraction.

Some of the procedures can take much time on a microcomputer. Once an implied edits routine ran for 2 hours and 28 minutes before it came back and said that it had run out of space. In error localization, where the system chooses fields for deletion in a succession of records, GEIS can get bogged down (but not stopped) on a single record. Some improvements can still be made on the speed. However, some of these procedures will still take several minutes which is not surprising when you consider that optimizing routines from operations research are being applied thousands of times in a row.

Data Review and Correction

After edits are written, they are applied and reports are written on the extent of the edit failures. Next, error localization is run to choose fields to be deleted. The minimum change rule is adhered to, that is, the fewest possible fields will be deleted. This is not always desirable, however, so the specialist can influence error localization by weighting fields according to their reliability.

The preferred method of imputation is hot-deck imputation in order to preserve multivariate distributions. Failing this, a choice of univariate or bivariate model based procedures are available. Some users might desire more flexibility in imputation. It does not appear possible to use imputation methods that are not provided. For example, in additivity edits, the minimum change rule might not be appropriate. If the edit $\mathbf{a + b + c = d}$ is failed where $\mathbf{d > 0}$, some specialists would rather that the values of **a, b,** and **c** be "scaled" so that their sum equals that of **d**. This does not appear possible in GEIS (though this might been done in a pre-edit).

The specialist sets up the imputation procedures in much the same manner as the edits are, that is, through ORACLE screens. Matching variables are specified as are choices of imputation methods. The mechanics of setting up the imputation specifications are quite easy.

Support, Updates, and Training

Documentation consists of one fairly good manual and several scholarly reports that can be read to get an idea of what the system is supposed to accomplish. The mechanics of using the system are well documented with the exception of jargon here and there that should be defined. As GEIS represents a departure from manual review, badly needed is an applications guide. A questionnaire of 45 or 50 variables should be used as an example of how an application might be programmed from start to finish. In this applications guide, there should be an indication of which edits should be applied in the pre-edit and which should be applied in GEIS. Also, there should be a discussion of how edit limits are set and how imputation options are chosen. In speaking with members of the GEIS team, this type of documentation is being written. Statistics Canada will not license the system to other agencies, at least in the near future. Those wishing to review the system and/or methodology may be able to secure a copy of the software if they are willing to provide feedback.

The current version of GEIS in Statistics Canada is version 6.2. Some screens and capabilities have been added and processing has been speeded up.

Summary

GEIS is a well-thought-out, well-developed system that a few in NASS have considered using for some surveys after an initial processing of data (assuming a license could be arranged). Before this could happen NASS would have to decide to use ORACLE as its database system on its newly acquired LANs in its state offices. Also, NASS would like to see GEIS used in production for a few large surveys in Canada to see how well it works.

SPEER

This system was evaluated in great part in the offices of the U. S. Bureau of the Census with the aid of members of the SPEER team. This was necessary because SPEER is undocumented and requires the assistance of a knowledgeable person to get applications running. For this evaluation, the application was programmed by Tom Petkunas of the U. S. Bureau of the Census while the author looked on and asked a lot of questions.

SPEER Vision

SPEER will model ratio edits because they are very common in economic surveys and because within themselves they are amenable to edit analysis and error localization. SPEER will make use of both batch processing and on-line review of forms. Where possible, use will be made of reported data. For example, responses are often reported in wrong units. In these cases, the reported answer will be converted to the proper units. It is not sought to eliminate the specialist from review of forms, rather this review will be structured around the Fellegi and Holt methodology. The specialist will review records based on certain criteria such as the size of the firm or on the changes made in batch. Full reports will be generated and the audit trail will be an integral part of the processing. Before SPEER is applied, preliminary editing and follow-up has been done in with another system.

How Productivity is Improved

The SPEER system does not necessarily reduce the hours of manual review though this may happen. Rather, the gains resulting from its application allow the analysts to review more forms within a deadline. SPEER addresses a particularly awkward processing situation where forms are collected, keyed, and pre-edited in Jeffersonville, Indiana, and undergo a "complex" edit in Suitland, Maryland. Under the old batch procedures, one cycle for a particular form could take weeks as the editor would have to obtain the form from Indiana. When it would arrive in Maryland, it could well have been a different editor who continued the review. With SPEER's combination of batch and on-line processing, this cycling has been eliminated.

How Quality is Improved

Quality is improved because each form is handled once by one editor. Full information is provided including the values of all variables and feasible regions for the important variables. Information is dynamically re-calculated, including new feasible regions based on newly imputed values of variables. An audit trail is an integral part of the edited data record and full reports are generated.

Types of Edits

The basic edit in SPEER is the ratio edit. Other edits, to the extent that they can be restated as ratio edits can be handled in SPEER. Associated with SPEER are satellite routines, while not part of the Fellegi and Holt methodology, can be used to state other kinds of edits. Additivity edits are often included there. Conditional edits are handled either by rewriting the edit into a ratio edit without the IF - ENDIF or by splitting the file. Many edits will have been included in the pre-edit conducted in Indiana. Another requirement is that all of the ratio edits be connected to each other. This requirement is usually no problem for most economic surveys but is not always satisfied in a few cases such as agricultural surveys. For example, the ratios

```
30 < corn bushels / corn acres < 180
5 < oats bushels / oats acres < 80
```

will not usually be connected together. Nevertheless, the SPEER team has found a way to apply the system to the Census of Agriculture.

Edit Writing and Edit Analysis

The mechanics of stating the ratio edit bounds are quite easy. The specialist provides a list of lower and upper bounds for each ratio. Harder to fathom is the way that the variables are organized in SPEER. They are classed as either "basic", "satellite", or "detail" items. The basic items are the most important and are edited against each other simultaneously through ratio edits. They are typically scattered throughout the questionnaire, being perhaps, the one or two most important variables of each section. Satellite items are second in importance and will typically relate to only one or a few of the basic items. The detail items are third in importance and represent an addend of a sum and the like. These can be involved in additivity or other consistency edits in satellite routines. These definitions are not set in concrete and a questionnaire can be represented in several different ways.

SPEER generates implied ratio edits which are inspected to determine if the data are being constrained in an unintended way. The bounds of the ratio edits are set through an iterative process which usually includes the testing of bounds on previous data. A statistical program in SAS has been written to facilitate this analysis. The programmer sets bounds for both a batch and an on-line manual edit. Actions taken in the batch processing can be overridden in the manual review. Unique to SPEER is an ability to widen bounds by the use of a multiplier which can be set either variable-by-variable or for a set of variables. Thus it is difficult to speak of levels of edit failure in SPEER. Also, SPEER has an extremal points capability though it is not called as such. The screen for the on-line part of SPEER generates bounds for each variable.

Additivity edits and the like which are written in the satellite routines are written directly into the system code in Fortran. The programmer must know both the system and Fortran in order to write these edits. These edits are not involved in edit analysis.

Survey Management

All callbacks and a pre-edit will have been conducted

before SPEER is applied. A batch run will be conducted on a mainframe where deletions and substitutions will be made. Generated as part of the batch processing are reports and an audit trail which will be available in the manual review. Forms are selected based on pre-set criteria such as size of firm or the character of changes made in the batch run. The manual review will be conducted on microcomputers, perhaps on a LAN. The editor will not require the paper form as all information is available to him. The form will be reviewed once and then stored.

System Items

SPEER is developed in modules which facilitates improvements. It is portable across architectures and operating systems. The use of Fortran as the base language provides great flexibility, particularly in the choice of imputation methods.

A screen has been developed so that subject matter specialists can more easily enter ratio edits. However further screens should be developed for all edit and imputation specifications. These screens should be tied together with a menu system so that the user does not have to know where in the system code various steps must be written.

The SPEER team spends a great deal of time customizing the system to various applications. SPEER does not enjoy the level of support as does Blaise in the Netherlands or GEIS in Canada. Much better would be an arrangement where the systems people would concentrate on developing the system and the subject matter specialists worry about the applications programming.

Data Review and Correction

SPEER does a rather neat job of tying the batch processing to the on-line manual review. This is accomplished by the way information is presented on the screen. For each of the basic items the corrected (current) and reported values are displayed as are a status code and the lower and upper limits of the feasible region for each variable. Two lines of the screen might look like:

Mnem	Corr	Reprt	st	lower	upper
femp	**1150**	**1150**	**r**	**492**	**1161**
fapr	**22780**	**15381**	**x**	**16172**	**23245**

If the variable is outside the feasible region, it is highlighted. A change made in value of a variable will cause the lower and upper limits of all the basic items to be recalculated. If the editor wishes to accept values which violate the edits then she can use the multiplier to widen the bounds.

In the example above, the change in the variable fapr could have been made either in batch or in the on-line review. It could have been made either through the application of one of a succession of model-based imputation options or manually.

Any model-based procedure can be coded into SPEER. However, the procedure is coded directly into the system code rather than through some user interface. Again, the assistance of the systems programmer will be needed. Donor imputation is not currently an option.

Movement in the screen and between screens could be improved. In order for the editor to get to a value, she must make a selection in a menu and type the name of the variable. Much better would be the use of arrow keys for navigating the screen. Another addition that would help would be the use of Pg Up and Pg Down keys in order to move between screens.

SPEER has the capability of modeling editor behavior as in an 'expert system', however, this was not tested for this evaluation.

Support, Updates, and Training

The systems aspects of SPEER are undocumented nor are manuals available. This plus the fact that edits and imputations are written directly into the system code make it very difficult to consider using outside of the Bureau of the Census. The methodology behind SPEER has been amply documented in a series of articles written by Brian Greenberg and his colleagues.

References

Cotton, P. (1988), "A Comparison of Software for Editing Survey and Census Data". Presented at the Statistics Canada Symposium 88: The Impact of High Technology on Survey Taking, Ottawa, Canada.

Fellegi, I. P. and D. Holt (1976), "A Systematic Approach to Automatic Edit and Imputation", Journal of the American Statistical Association, Volume 71, Number 353, Applications Section, 17 - 35.

Hanuschak, G. et. al. (1990), "Data Editing in Federal Statistical Agencies". Statistical Policy Working Paper 18, Subcommittee on Data Editing in Federal Statistical Agencies, Federal Committee on Statistical Methodology, Office of Management and Budget, Washington, D. C. (contains editing checklist and glossary of editing terms).

Pierzchala, M. (1988), "A Review of the State of the Art in Automated Data Editing and Imputation". NASS Staff Report No. SRB-88-10, USDA, Washington, D. C.

Footnotes

1/ Blaise rhymes with fez. Blaise is the first name of the mathematician Pascal whose name was taken for the programming language in which Blaise is written.

2/ CATI = Computer Assisted Telephone Interviewing, CAPI = Computer Assisted Personal Interviewing.

3/ Reviewed by Rita Hohenbrink, formerly of NASS, now with the Department of Energy.