15
Chapter

Edit Authoring Techniques

Chair: Jim O'Reilly, Research Triangle Institute

Shirley Dolan

J. Tebbel ◆ T. Rawson

Robert F. Teitel

Methods of Reusing Edit Specifications Across Collection and Capture Modes and Systems

Shirley Dolan, Statistics Canada

15 Chapter

Abstract

As multi-mode approaches become a viable means of collecting and capturing data, the issue of reusing edit specifications across modes and software systems becomes an important issue. Where more than one system has to be used in order to offer the respondent a choice of reporting methods, statistical agencies are faced with the problem of having to develop and maintain more than one set of edit specifications, that is, one for each mode or system used.

This presentation explores some of the methods currently being used and others which are being discussed at Statistics Canada, as solutions to the problem of reusable edit code. Among the ideas are:

- multi-mode options offered by DC2, Statistics Canada's primary collection and capture software, which includes reusable edits, and
- the potential of using DC2's editing engine with other systems used at Statistics Canada to collect data using laptops and electronic questionnaires.

Exploiting the increasingly automated traditional collection and capture modes as well as the emerging methods either singly or in combination promises to deliver savings in resources and improve timeliness and data quality. This promise is unfortunately offset by the increased difficulties in developing and maintaining different versions of the edit specifications, when more than one system is used within a survey or where there is a requirement to apply edits at different stages of the process. This paper first explores the various collection and capture methods available today followed by a discussion of typical automated editing procedures. Finally, some existing and potential solutions to the problem of maintaining multiple versions of the edit specification are presented.

Methods of Reusing Edit Specifications Across Collection and Capture Modes and Systems

Shirley Dolan, Statistics Canada

Multi-Mode Collection and Capture

In the mid-1980's, multi-mode (or mixed-mode) collection and capture was typified by "surveys which combine the use of telephone, mail, and/or face to face interview procedures to collect data for a single survey project" (Dillman and Tarnai, 1988). This is still the most popular view, but our approaches have become much more automated. Telephone surveys are frequently done with the assistance of a computer. Personal interviews can now be conducted with the aid of laptops. And newer mechanisms for processing paper questionnaires, such as Intelligent Character Recognition (ICR) technology, promise to improve the timeliness and quality of data collected by mail. In addition, other modes are beginning to receive increased attention.

Perhaps the most interesting of these is the idea of the respondent extracting data from their MIS and sending it in electronic format to the statistical agency. In fact, this method is not new and has been used at Statistics Canada since at least the early 1970's. As larger respondents (such as provincial governments) invested in computer automation, it became feasible (and desirable) to receive data on a tape destined for the mainframe. There were a number of problems associated with this method. It took considerable time to negotiate a file format and content with each reporter. Despite the best efforts to standardize, it was often impossible or not cost effective for the respondent to conform exactly to the proposed formats and code sets. This added to the development and maintenance burden at the Bureau when custom programs had to be built to read and process the differing formats. Processes tended to be batch-oriented with data being stored on flat files and this added to the effort and time needed to prepare the data tapes. However, despite these constraints, electronic reports did not fall out of favour and new technologies such as improved programming languages, database management systems and advanced communication methods have significantly improved the potential usefulness of this mode of reporting.

Another mode which has been used at several statistical agencies is the electronic questionnaire. Although there are many variations on this theme, this approach generally consists of the development (and maintenance) of an interactive questionnaire with on-line help and edits which is copied to diskette and mailed to the respondent. Data is entered using the software and then returned via the diskette. There are many advantages to this method. The paper questionnaire is eliminated, edits are performed in the presence of the respondent, and the data arrives at the statistical agency already in electronic format and, at least to some degree, edited. In some cases, incentives to use this method of reporting are built into the software, such as data manipulation and reporting features which are attractive to the respondent. The disadvantages to be considered are the development and maintenance costs,

which can be significant particularly when added-value features are built in, the additional processing requirements (diskette generation, reception and archiving of diskettes, decryption and virus scanning to name a few) as well as potential liabilities resulting from respondent expectations for the extra features included in the software.

A variant of the electronic questionnaire is the notion of providing a questionnaire to the respondent over the Internet. This can be done in any number of ways. For example, it is possible to develop a questionnaire using Hypertext Markup Language (HTML). This type of questionnaire or form is accessed by the respondent using the World Wide Web services. The data is sent to the statistical agency using E-mail. There is at least one commercial product called Decisive Survey, from Technology Corp. (Chrisholm, 1995) which also uses the E-mail/Internet approach. This product boasts a drag and drop method to questionnaire development, runs on Windows 3.1 or Windows 95 and interfaces with several E-mail systems.

Although there are many other methods of collecting and capturing data such as pen-based computers and touch-tone technology, this analysis will focus on those mentioned above.

Data Editing

Data editing is a vast topic which can include not only the basic editing which takes place during collection and capture, but also during other stages of the survey process such as inter-case editing for imputation purposes or editing to detect outliers. Editing can also be described as either manual or automatic. For the purposes of this discussion, automated editing is assumed; that is editing which takes place either interactively or in batch, in the context of a computer program or system. Also, editing refers to the checking of data applied during the collection and capture phase. These are typically:

- ***Preliminary Edits*** -- usually used to detect gross reporting or keying errors at the field level. Validation of format types, range checks and verification of simple code sets are commonly included in this category.

- ***Consistency Edits*** -- inter-field value checking, may include computations to ascertain conformity.

- ***Historical Edits*** -- a variant of the consistency edit, where the values for the most recent report are compared to those of previous reports. This typically involves comparison of gross differences against tolerance tables. For example, a warning may be issued if the value reported for number of employees this month differs from what was reported for last month by more than 10 percent.

The requirements for data editing can be different depending on the mode and stage of the collection and capture process. Consider the following illustrations.

First, editing which is applied during a personal interview using a laptop may not include complex consistency or historical checks. These may be applied at a central site where reference files and previously reported data is stored and where more powerful processing equipment is available. It is not

unusual to re-apply the preliminary edits done in the field following the application of the more complex and subjective edits to ensure that these have not compromised the basic edit rules. In this scenario, separate versions of the preliminary edits are required for the laptop application and the centralized editing process.

Second, offering respondents a choice of reporting modes increases the potential for the need to develop separate versions of the edits. A large economic survey could, for example, have respondents who report by paper questionnaire, by telephone and by electronic questionnaire. The paper questionnaires are processed using a system designed to accommodate rapid data entry. A second system, offering Computer-Assisted Telephone Interviewing (CATI) features such as call scheduling, call outcome coding and calling protocols is used to collect data from respondents preferring to report by telephone. The applications built using these two systems would have their own separate versions of the edit specifications. A third version of the edits would be included in the electronic questionnaire.

A third scenario is one in which respondents transmit data extracted from their database. The edits applied to this type of response should be minimal but it may be necessary to apply consistency or historical edits. Typically, electronic data is edited (in a batch process) with edit exceptions being addressed interactively. There is the potential to require different versions of the edits for the batch stage and the interactive stage.

The preceding examples illustrate scenarios in which the potential exists for different versions of the editing specifications to be developed. Edit specifications are an important part of any collection and capture process and their development and maintenance consume a significant percentage of the resources expended in building and testing an application. This is of course multiplied when it must be done for each mode or stage of editing used. Further complexity is added when the systems used differ in their support for specifying an edit. It may be that an edit written in one language or syntax cannot be represented to the same extent in the other system used. DC2, Statistics Canada's generalized collection and capture system, offers a attractive solution to these problems.

Reusing Edit Specifications in DC2

DC2 provides support for mixed mode processing, that is, data reported by questionnaire, by telephone and in electronic format. With DC2, edits may be specified once and reused within a survey to validate data, regardless of the method used to report the data. And there is considerable flexibility in the application of an edit. Although the same piece of code will be called to edit a value (or set of values), there are choices which can be made in how the edit is applied and actioned. In other words, the edit behaviour can be tailored to the individual needs of the particular mode being used without the need to have a separate version of the edits for each variation. The following example will illustrate this.

Data reported on questionnaires can be captured in at least two ways:

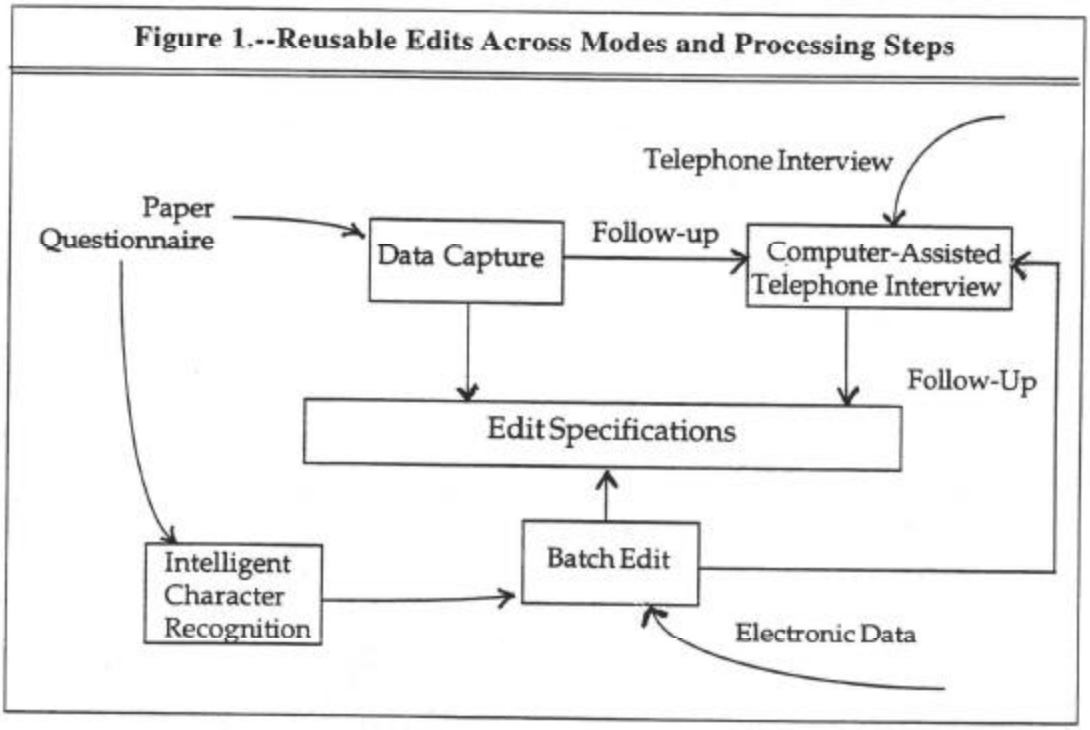
- ***Heads-Down Keying*** -- experienced key operators capture the data from the questionnaires. The term heads-down refers to the practice of keeping the eyes on the questionnaire while keying the data. The emphasis is on high key-stroke rates with low keying errors. Consequently, the capture operation is normally only interrupted when a potential keying error (identified by an edit) is detected. Reporting errors are typically corrected at a second stage by editors who are familiar with the subject matter.

- ***Intelligent Character Recognition*** -- the questionnaire data and image are captured by machine. Basic format edits are applied to detect interpretation errors. As with the operator-captured data, the subject matter corrections are normally done by experienced editors.

With DC2, the suite of edits is applied interactively and the results (passed/failed) are stored as the questionnaires are key-entered. ***However, the key operator is advised only of those errors which indicate a keying error.*** The machine-captured data is loaded into the DC2 database and edited in batch, again using the same suite of edits applied during the heads-down keying exercise. Correction of errors identified from either mode would typically be done through a computer-assisted telephone interview. The previously identified edit failures would be listed for the interviewer, corrections would be made as indicated by the respondent and the edits would be reapplied, interactively, as changes were made to the data. ***At this stage, the interviewer is advised of all edit violations.*** So, although the same edits are used, the behaviour surrounding their application is tailored to the mode at hand. It is even possible to build a generalized edit which uses different reference files (or some other variation) depending on the collection/capture mode.

This example, depicted in Figure 1, illustrates the flexibility of the editing facilities in combination with the desirable feature of being able to reuse the edits across modes and various stages of the collection and capture process.

Figure 1.--Reusable Edits Across Modes and Processing Steps



Many variations on the above theme are possible:

- A survey in which initial collection is shared by mail-out/mail-back and CATI. CATI is used for follow up.

- A survey in which some respondents report on an electronic questionnaire and some by mail. The electronic questionnaire data is treated as an electronic source. It is loaded into DC2 and edited in batch.

- A survey in which personal interviews are conducted mainly using laptops with some exceptional personal interviews done with paper-and-pencil method. As in the previous example, the output from the Computer-Assisted Personal Interview (CAPI) is edited in batch.

In the last two scenarios, DC2 can improve consistency of editing approach between the paper questionnaire and the electronic report by having the standard set of questionnaire edits applied to the electronic source. This would identify inconsistencies in the two sets of edits (the field edits and those applied using DC2) and would provide a centralized repository for any complex edits not included in the field edits.

DC2's mixed-mode support has, to a large degree, met the challenge of reusing edit specifications across collection and capture modes. This is certainly true for the more popular modes (paper questionnaire and CATI). The ability to process electronic data extends the reusability of the edits to other modes, if not at the point of capture, at least at the back-end as a centralized repository, editing and follow-up facility. However, the potential exists to extend the mechanism. It is conceivable that the editing facilities could be extracted from DC2 and made available to other external collection and capture systems. Before exploring this possibility, it is worthwhile looking at the DC2 editing environment.

The DC2 Editing Environment

The DC2 editing environment (Statistics Canada, 1992), shown in Figure 2, includes the following three main components:

- A specification language
- A compiler
- A run-time engine.

The ***specification language***, known locally as the Edit Specification Language (ESL), is based on the Prolog programming language. Some extensions have been made to accommodate the special case of editing statistical data. For the following reasons, Prolog is an ideal paradigm on which to base an editing language:

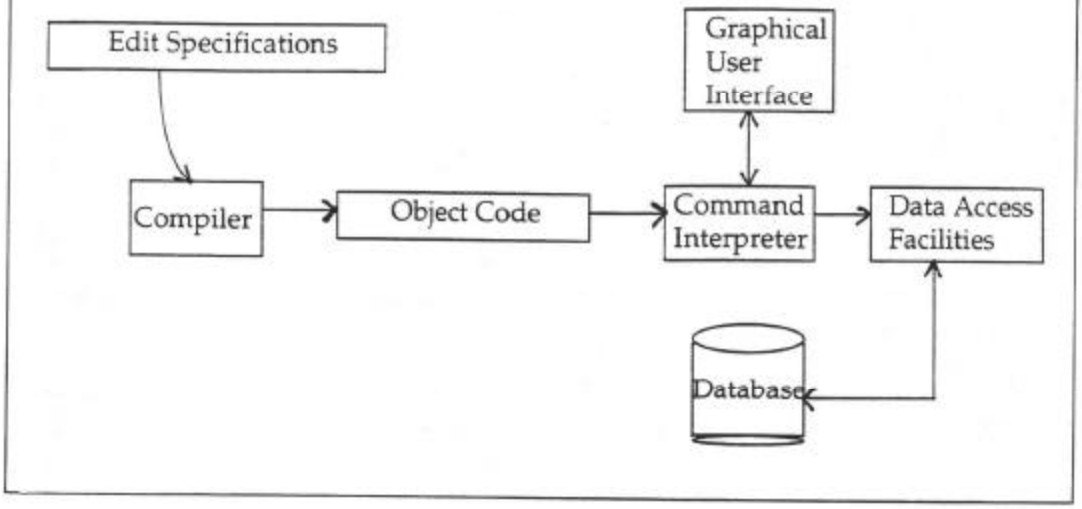
- It is a rule-based language with built-in pattern matching and backtracking.

- Prolog's concept of failure matches well with the concept of edit failure and exception handing.

- With a small set of extensions, numeric edits can be easily expressed.

- Prolog code can be compiled using a technology known as the Warren Abstract Machine (Ait-Kaci, 1991) that is in the public domain.

Figure 2.--The DC2 Editing Environment



The ***compiler*** is a Prolog program, developed at Statistics Canada, which is currently running under ALS Prolog from Applied Logic Systems. The compiler reads the ESL source code and generates code which can be executed by the run-time engine.

The ***run-time engine***, also called the command interpreter, is an in-house program which is a close approximation of the Warren Abstract Machine. It functions as a virtual computer that reads, interprets and acts on the instructions in the form of assembled (and compiled) ESL. The program is written in ANSI C and is, therefore, potentially portable to any computing platform. Within the DC2 system, it operates in two ways:

- As part of the overall production engine: receiving data via the capture instrument or the database, applying the specified rules, and reporting the results back to the calling program which subsequently stores them (and the data) in the database and/or displays them on the screen, and
- As part of a testing/debugging tool: receiving data from a programmer via the command line, applying the specified rules, and reporting the results back to the programmer.

The ESL will, of course, execute anywhere the command interpreter is available. So it becomes a matter of making it available where needed. Could this editing facility be incorporated into other collection and capture systems? This idea is visited next.

Using the DC2 Editing Engine in Other Systems

Naturally, the question of whether the DC2 editing facility can be incorporated into other collection and capture systems depends on the potential of the editing engine to interface with these systems plus the ability of these systems to incorporate a foreign editing mechanism. For this discussion, we will address systems running on laptops and those used to develop electronic questionnaires.

Could other systems dynamically access the command interpreter at run-time? The system in question would have to be designed to allow user exits or some similar mechanism to interface with external code. Most systems today allow some type of user exit -- this is commonly an interface to one or more established programming languages. Two scenarios for integrating the DC2 editing environment into a commercial product come to mind.

In the first instance, the system could communicate directly with the editing engine. This would require having the third party product extended so that it could interface with the DC2 command interpreter. This might not be attractive to the supplier of the product given that the applicability of the mechanism would be limited and of interest only to Statistics Canada. A second and more attractive approach would be to have the third party product communicate with the command interpreter through a non-proprietary interface such as the C programming language or Visual Basic. Many software products offer user exits to C or some other equally suitable programming language. It would work something like this (see Figure 3).

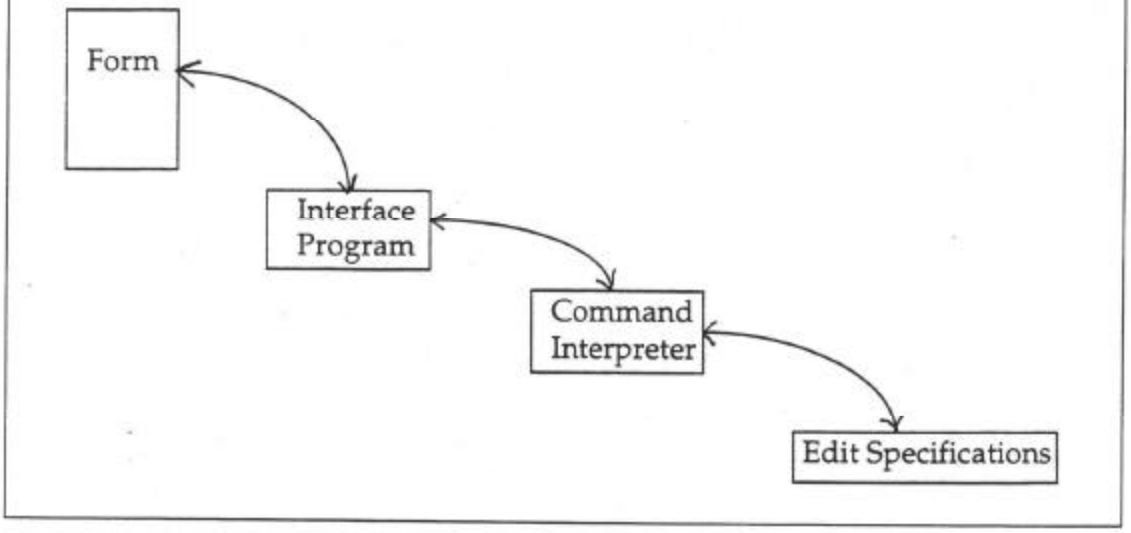
- A collection and capture application form is developed for a laptop or electronic questionnaire using a commercial product.
- At certain points during the capture exercise (when a field value is entered or values for a collection of fields are available), the application issues a user call to an external routine.
- The external routine accepts the data value or values and passes them to the DC2 editing engine which applies the ESL code for the field or fields in question.
- The data is validated and the results are returned to the application via the external routine.

An interesting variation to this approach is the idea of accessing the editing engine to perform checks on data entered into an HTML or Java electronic form. Data entered into the form are processed using a technique called Common Gateway Interface (CGI). A CGI program could call the DC2 editing engine (and the ESL code) and return any edit violations to the user. For reasons mentioned below, this is likely the most feasible use of the DC2 editing environment.

The other side of this equation is the potential of the editing engine to be incorporated into or called from external systems. Some considerations are:

- The ESL can be executed wherever the command interpreter exists.
- The command interpreter is designed to be part of a system; that is, it is designed to be called (accessed dynamically at run-time).

Figure 3.--Accessing the DC2 Editing Engine from External Systems



- The interface requirements are simple and well defined. Essentially, there are two types of edit interfaces: one for single field editing and one for editing two or more fields.

- Data access methods (to an Oracle database) are restricted to discrete routines which could be replaced if required to access some other data storage mechanism.

- The most likely computing platform for laptop applications and electronic questionnaires is Intel-based and running some flavour of Microsoft Windows or Windows NT. Written in ANSI C, the command interpreter could be ported to these platforms. However, this should be considered a non-trivial exercise. The interpreter was developed for UNIX and there are some major differences which would require careful retooling. For example, the UNIX and Windows TIME routines operate differently. TIME is an important element in any statistical editing mechanism. Byte ordering is also different on these two platforms. These types of restrictions suggest that the DC2 editing facilities could be more easily and more cheaply adopted for wider use on UNIX platforms.

Conclusions

For the majority of mixed-mode applications, the DC2 system offers a solution to the problem of having to write and maintain separate versions of edit specifications. Whether the method of collection/capture is by paper questionnaire, telephone interview or electronic data report, an edit can be defined and maintained at the survey level and used seamlessly across reporting modes.

The ability of reusing the edit specification does not restrict the possibilities for choosing relevant behaviour based on the mode of collection. For example, error messages can be reported to the user when appropriate.

It is technically feasible to use the DC2 editing engine in other external collection and capture systems. However, feasibility should not be confused with desirability. The task of porting the engine to a non-UNIX platform may not be cost-justifiable. Commercial systems may not be sufficiently "open" to incorporate DC2's editing facilities and the editing facilities may have platform dependencies which make it difficult to move from the UNIX environment to the Windows platform. Statistics Canada will be evaluating these possibilities and others in its pursuit of methods and techniques for reducing the instances of edit specifications required by applications.

References

Ait-Kaci, Hassan (1991). Warren's Abstract Machine, Massachusetts Institute of Technology.

Chrisholm, John (1995). Surveys by E-Mail and Internet, *Unix Review*, pp. 11-16.

Dillman, Don A. and Tarnai, John (1988). Administrative Issues in Mixed Mode Surveys, *Telephone Survey Methodology*, John Wiley & Sons, Inc., pp. 509-528.

Statistics Canada (1992). ***DC2** Edit Specification Language Reference Manual.*

Chapter 15

CDC Edits: Tools for Writing Portable Edits

J. Tebbel and T. Rawson,
U. S. Centers for Disease Control and Prevention

Abstract

The CDC EDITS Project has produced a system for writing collections of executable data edits, which can be distributed as part of a public standard. These collections of edits can be used by interactive data entry programs to achieve real-time field-by-field validation or in batch processes for data already collected.

EditWriter is a complete menu-driven development environment for creating, maintaining, testing, and documenting data edits. Individual edit checks are written in the EDITS language, a C-like language with simplifications and extensions for the editing task.

EditWriter is capable of creating and manipulating all the structures needed to test data: code snippets, data dictionaries, record layouts, and reference tables. The output of EditWriter is an object called the "Metafile."

The EDITS Engine is an interpreter that processes the Metafile when called by an application program to test a field or record of data. It is supplied as C-language source code that can be compiled and linked on a variety of computing platforms or as a Dynamic Link Library (DLL) for use with most database packages in the Windows environment.

EDITS Metafiles have been used since 1993 to improve the quality and efficiency of processing in CDC's national Behavioral Risk Factor Surveillance System. National standard-setting organizations for cancer registry data have adopted EDITS and are currently distributing Metafiles.

CDC Edits: Tools for Writing Portable Edits

J. Tebbel and T. Rawson, U. S. Centers for Disease Control and Prevention

Introduction

The Centers for Disease Control and Prevention (CDC) created the EDITS system to improve the quality of data collected by cancer registries. CDC's Division of Cancer Prevention and Control administers the National Program of Cancer Registries (NPCR) authorized by Public Law 102-515, which was passed in 1992; the program's goal is to help establish new state cancer registries and to update existing ones. Some of the existing registries have collected many years of data in a variety of existing systems, but there is no agreed-upon standard for data checking. This shortcoming impairs the use of the data, a real concern as researchers seek more information about cancer and how to prevent or control it.

CDC's EDITS is a collection of computer programs and data objects. These software tools were intended to encourage independent authorities, who sometimes have competing interests, to contribute to and accept voluntary, shared public standards for data quality. EDITS was also intended to provide the means for efficient development, testing, documentation, and publication of standard data checks in an executable form. The system is neither cancer- nor health-specific; it can be used for any type of data on a variety of hardware platforms and in diverse operating system environments.

Edits as Quality Assurance

Even when data collectors intend to adhere to a standard, the details of field-by-field checking often vary according to the decisions made by individual programmers. The EDITS system eliminates this source of variability by producing a portable, executable version of data-checking logic as specified by the authority for a standard. The data object, which contains an expression of the validation rules, can then be distributed for direct execution upon files and records of data in a variety of processing scenarios. The same edits can be applied at different points in the flow of data through a system; data already collected can be checked in batch mode, and new data can be tested as they are being entered. This feature makes it easy to integrate EDITS into existing systems; processors can apply existing standards in batch mode with very little cost or disruption to operations. When the correction of errors will be costly, identical edit logic can be attached to data entry programs to catch mistakes when they are most readily corrected.

Setting and implementing data standards is not without potential problems, as detailed in Figure 1. These problems are not necessarily completely solvable with software, but portable edits provide a good beginning.

Figure 1.--Some Advantages to the EDITS System

Problem	How EDITS Addresses
Multiple organizations may set and revise conflicting or overlapping standards for the same data item.	Supports consensus-building among standard setters and enables collaboration by showing differences.
Programmers interpret each standard and render it into code for individual system	Standards are directly executable and portable across languages and platforms to avoid reinterpretation.
Data collected differently became different data and may not be comparable.	Data checked by the same edits may be comparable. Difference in edits serves as documentation of how data differ.
Adopted standards are sometimes adapted for local needs or ease of processing.	Availability of standard in executable form may eliminate need for adaptation.
Only simple edits implemented in interactive mode, more complex cross-field checks saved for batch mode.	Same edits can be used for interactive and batch mode.
Standard sometimes buried deep in source code. Documentation sometimes missing, or out-of-sync with edit logic.	Standard separated from application source code so it can be developed, tested, and maintained separately. Documentation can be kept with logic. Can generate reports with logic and documentation.

Cancer Registry Specifics

Cancer registries may report data to one ot more of the following: the NCI SEER program, the North American Association of Central Cancer Registries (NAACCR), or the National Cancer Data Base (NCDB). The data are submitted in a standardized format, but data users need assurance that each field was collected uniformly across all data collection activities. Currently, cancer registry applications are in use on multiple platforms, including MS-DOS, Windows, UNIX, and VMS. As many of these applications implement (at least partially) existing standards, the solution is not necessarily to have additional standards. As a user of data from all of these sources, CDC facilitated the development of EDITS to provide a better means of expressing and using data standards, with the ultimate goal of improving data quality.

Development of EDITS System

The EDITS System was developed with input from competing cancer registry software providers.

In 1991, development began at CDC with a rapid prototype of a linkable C language interpreter module. The concept of having portable edits that could be used anywhere the C interpreter could be compiled was tested and proven with exploratory programming. Performance during this early test was adequate for interactive processing of a record or a few fields at a time but required enhancement for batch processing of large files. Over time, the language evolved slightly from C to make the edit logic

more readable and add extensions specific to data editing. The final design addresses performance issues by replacing the C interpreter with a compiler and p-code interpreter for faster edit execution and by adding indexes for faster lookup table access.

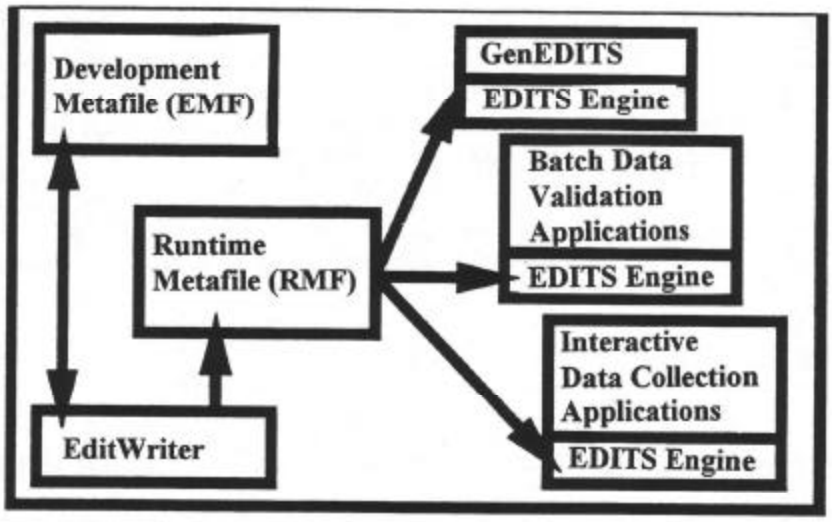
Components of EDITS System

The EDITS Metafile is a single file that encapsulates internal databases. There are tables for the data dictionary, record layouts, edits, and error messages; there can also be any number of lookup tables for table-driven edits. The Metafile has both development and compiled run-time formats; the run-time Metafile is portable across architectures.

The EditWriter is an integrated development environment for maintaining a Metafile that can create and maintain data dictionaries, define record layouts, write and interactively test edits, and create and import tables. The present version of EditWriter is an MS-DOS application written in C and FoxPro.

Edits are written in the EDITS language, which is based loosely on C with a rich function library for editing data. It is compiled to p-code to obtain a mix between speed and portability. Edits may be thousands of lines in length, or just two or three.

The EDITS Engine accesses run-time Metafiles to execute edits and return error messages. It is called via the EDITS Application Program Interface (API) and is callable from C for MS-DOS and other platforms and as a Windows DLL for use by any Windows language. There are options available for edit execution, including SKIPFAIL for skipping multi-field edits where any single field has failed and SKIPEMPTY, which skips edits where any field is blank.



This diagram shows the relationship of the different parts of the EDITS system. EditWriter is used to maintain a Metafile and compile it to a run-time Metafile.The EDITS Engine is then called by batch and interactive applications to perform edits from the run-time Metafile. Multiple applications can use the same run-time Metafile.

Driver Programs Incorporate the EDITS API

An EDITS driver is any program that incorporates the EDITS API. In batch mode, a driver program can detect existing bad data. In interactive mode, edits can prevent bad data. EDITS Drivers can be written in C or in any Windows language

GenEDITS is a generic EDITS driver that works in batch mode only. It produces a report of errors encountered, including the name of the edit, error message, and a list of fields referenced in the edit. GenEDITS also includes a summary report of failure count by edit. It can be used for recoding or reformatting data and to calculate simple frequencies.

Current Uses

Most EDITS users are using MS-DOS platform in batch mode. CDC's Behavioral Risk Factor Surveillance System has been using EDITS at both the state level prior to data submission and at CDC since 1993. The NCI SEER program is now maintaining its edits both in the original COBOL and in EDITS, which gives it a portable solution. The EDITS system consistently gives the same results. NAACCR recently released a Metafile of cancer edits incorporating standards from SEER, the American College of Surgeons, and others.

EDITS is available at no charge and may be obtained by downloading via anonymous ftp (address: ***ftp.cdc.gov, path: /pub/Software/EDITS***), World Wide Web (address: ***http://www.naaccr.org***), or by contacting the authors. ■

15
Chapter

Skip Patterns and Response Bases: Graph Manipulation in Survey Processing

Robert F. Teitel, Abt Associates and George Washington University

Abstract

By incorporating explicit indications of skip instructions (or GOTOs) in a survey description language (which also includes the usual facilities for the question identification, question text, response values, recodes, etc.), it is possible to build an acyclic directed GRAPH of the data collection instrument.

The survey-derived graph -- with questions as nodes and skips as edges -- may be manipulated using graph algorithms to prepare the response bases for each question for display in the printed codebook, and to prepare the control information for a skip pattern editor for use while processing the actual data.

This talk will describe these processes and illustrate some of the results. ■