

An alternative approach for the implementation of data editing: the INSPECTOR project

Gregory Farmakis,
Agilis SA, Athens, Greece

José Figueiredo,
Instituto Nacional de Estatística, Lisbon, Portugal

Daniel Mota,
Instituto Nacional de Estatística, Lisbon, Portugal

George Petrakos,
Panteion University and Agilis SA, Athens, Greece

Daniel Santos,
Instituto Nacional de Estatística, Lisbon, Portugal

Photis Stavropoulos,
Agilis SA, Athens, Greece

Summary

The subjects of this paper are an alternative approach to data editing devised by the INSPECTOR project and the software prototype developed in order to implement it.

The novelty of the approach lies in the treatment of edits as definitions of the domains of variables and not as Boolean statements of relationships between variables. Users simply declare the variables (and their domains) required in order to implement a set of data edits instead of writing computer code. A suitable database stores these definitions for future use. An important point is that the variables are a superset of the survey variables, comprising additional variables defined as functions of survey variables and as auxiliary or historical variables or as vectors of other variables.

The prototype has been designed in such a way as to enable the implementation of several installations which share edits between them. Such an installation, in Eurostat and national statistical offices for example, can promote the harmonisation of editing practices.

Keywords: data editing, data validation, implicit variable, composite variable, domain-based validation, enhanced dataset, dataset template, abstract data source

1 INTRODUCTION

Large amounts of statistical data are regularly collected for various purposes, from various sources and with various methods. They unavoidably contain errors in the form of missing, unusual or logically inconsistent values. Statistical data editing is the process whereby statistical data are examined for errors and the identified errors are

corrected. In practice it is not always possible to correct errors. Editing attempts to at least modify the data, as little as possible, in such a way as to render them complete and logically consistent. The resulting 'clean' data can then be subjected to statistical analysis.

Moreover, editing is important for the producers of statistical information (statistical offices, government agencies, international organizations, etc.) because it helps them to improve their performance: the rationale behind this is that investigation of the causes of errors in the data, spotted by editing, identifies 'flaws' in survey operations and the correction of the flaws improves future performance. Data editing in this sense becomes an indispensable tool for Total Quality Management; see, for example, Granquist (1997), Engstrom and Granquist (1999) and Nordbotten (1999).

We can distinguish two 'kinds' of editing. Micro editing is concerned with the examination of each data record on its own or of its consistency with other specific records (e.g. records for related persons within a household). Macro editing, on the other hand, examines data records in relation to a large number of other data records. Outlier detection is the main type of macro editing; a record is examined for proximity to the main bulk of collected data.

The theoretical foundations of micro-editing, and more specifically of within-record editing, were set by Fellegi and Holt (1976). The Fellegi-Holt method constitutes an elegant proof of which fields' values must be modified so that a record is made to satisfy all edits. The problem of the method is that it still is computationally very demanding and that it is mainly suited to categorical variables and to specific types of numerical edits only. A long of research effort has been expended on devising numerical methods in order to speed up the method and to make it applicable to all kinds of edits. Some key references are: Sande (1979), Garfinkel *et al* (1986), Schopiu-Kratina and Kovar (1989), Fillion and Schopiu-Kratina (1993), Winkler (1998), Chen (1998), Quere (2000), Winkler and Chen (2002), De Waal (2003) and De Waal and Quere (2003).

Several software systems have been developed in order to carry out editing in large organisations. Most of them use the Fellegi-Holt method, either in full or, more usually, in approximate form. Some examples are CANEDIT and GEIS (Kovar *et al*, 1991) in Statistics Canada, DISCRETE (Winkler and Petkunas, 1996) and SPEER (Draper and Winkler, 1997) in the U.S. Census Bureau, CherryPi (De Waal, 1996) in Statistics Netherlands, AGGIES (Todaro, 1999) in the U. S. National Agricultural Statistics Service, SCIA in Italy and DIA in Spain. To these systems we should add Statistics Canada's CANCEIS (Bankier, 2003) which implements a slightly different approach than Fellegi and Holt's, the New Imputation Methodology (Bankier, 1999).

In this paper we present an alternative approach to data editing and a respective alternative computational approach. The motivation behind this approach was the desire to find a way of applying editing which on one hand is nearer to the logic of database management systems and on the other would enable the easier sharing of edits between organisations. This approach was investigated by the INSPECTOR project and for this reason we begin with a brief presentation of the project.

1.1 THE INSPECTOR PROJECT

INSPECTOR is a research and development project, partially funded by the European Commission under the ‘Information Society Technologies’ program (contract number IST-2000-26347). It was initiated in July 2001 and finished its activities in December 2003 (total duration: 30 months). The project comprised six partners, representing a mix of competences with a fairly wide geographical coverage. More specifically, the consortium consisted of two university departments, two national statistical offices and two private research and development companies, from Austria, Greece, Italy and Portugal.

The project concentrated exclusively on data validation, that is the checking of data for errors. The strategic goal of the project was the design and development of a generic, distributed and flexible data validation system, which would be able to be seamlessly integrated in the current (or future) processes of statistical data collection. The system should be accessible in a distributed way in order to validate large statistical data sets before their transmission or even throughout their production. From the outset it was considered that the usual declarative approach to validation (i.e. the specification of edits as “IF-THEN-ELSE” statements in a computer language of choice) does not facilitate interoperability of validation software or maintenance / sharing of edits. Therefore, an alternative approach was proposed for investigation and implementation. This is the approach that forms the core of the paper and is presented in a subsequent section.

During its lifetime, the project analysed the edits used in several real surveys by the two national statistical offices participating in it. The fit of the proposed approach to the editing practises of the two offices was subsequently tested. The approach was refined accordingly and a software prototype was designed and developed. Finally, the prototype was tested on real survey data for the implementation of real edits. Some of these results are reported in the present paper.

More detailed information on the project may be found in its homepage, www.liaison.gr/projects/inspector/.

1.2 ORGANISATION OF THE REST OF THE PAPER

The rest of the paper is organised as follows. Section 2 presents in detail the approach to data validation advocated by the INSPECTOR project. It is followed by a presentation of the software prototype built to implement it, in section 3. The approach and software prototype are demonstrated in section 4, in an application to real survey data. The conclusions of the paper are given in section 5.

2 THE INSPECTOR APPROACH TO EDITING

In this section we will present in detail the approach to data editing adopted, investigated and refined by the INSPECTOR project.

However we should make clear from the outset what is the scope of this approach. As mentioned in section 1.1 of this paper, the INSPECTOR project does not deal with all the steps of a complete data editing process. Its focus is on ‘data validation’. By this term we denote the checking of statistical (or other kind of) data for missing values, errors and logical inconsistencies and the notification of the responsible personnel for

any errors found. Therefore the approach that we will present does not cover error localization (identification of the fields which must receive new values in order to satisfy all edits) or error correction.

In the remainder of this paper we will speak of data validation. Data edits will respectively be referred to as ‘validation rules’.

2.1 DOMAIN BASED VALIDATION – THE INSPECTOR APPROACH EXPLAINED

Every validation rule is a statement about one or more variables and about a relationship between them, which must be satisfied. Rules are usually translated in computer code with the use of “IF – THEN – ELSE” statements.

The defining element of the INSPECTOR approach is **domain-based validation**. INSPECTOR views each validation rule as the declaration of the domain of definition (domain henceforth) of a variable or of a combination of variables. To give some examples, Income is a positive integer number, Economic activity takes its value among valid NACE codes, the difference between’s Mother’s Age and Child’s Age exceeds a certain lower limit, the combination of Age and Marital Status takes certain values, etc.

In other words validation rules define domains of individual variables and place restrictions on the Cartesian products of individual variables’ domains. The approach of INSPECTOR is that validation rules should be declared to and stored by a validation system as sets of variables and their domains and not as statements of relationships. In this way, validating a dataset will simply mean checking whether variables take allowed values or not.

The advantage offered by this approach is that the validation routines are generic and need not be re-written for each new application; users just need to declare the variables and their domains for each different validation setting. Adopting this approach however, demands from users the change of their frame of thinking about validation and the use of a number of novel concepts. We now proceed to describe these concepts in detail.

We assume that a survey has produced an amount of data stored in a dataset, which we call the **primary dataset**. This is the dataset that must be validated so that, afterwards, ‘errors’ are ‘corrected’ and statistical analysis is applied to its data. The records of the dataset¹ may be of more than one record types, representing different kinds of statistical units.

In an imaginary Census dataset, which we will be using as a recurrent example, we will have a record type for dwelling data, another record type for household data and another one for individual person data. Moreover there may be a metadata record as a prefix to the dataset. Records of different types may be related: each dwelling record will have household records related to it (for the households that live in the dwelling), which in turn will have person records related to them (for the persons comprising the households). Therefore there is a hierarchy of statistical units reflected in a hierarchy

¹ Dataset also denotes the computer file containing the data.

of record types. The metadata record type is the highest in the hierarchy since it refers to the whole survey.

Natural groupings of records form observations. In the Census dataset for example, each observation is the amount of data from a single dwelling and it therefore comprises a dwelling record, one or several household records and one or several person records for each household record.

Each validation rule refers to a particular statistical unit; the record type that corresponds to it is the **rule's base record type**. The rule may, however, involve variables from other record types too. Suppose for example, that in the Census, the household record type contains variable Total Household Income, while the person record type contains variable Income. A rule which states that '*Total Household Income should be equal to the sum of Income for the persons that comprise the household*' clearly refers to the statistical unit household. Its base record type therefore is household, but it also involves variables from record type person.

Every rule resolves to the definition of a variable and its domain. The variable may be univariate (such as the difference between Total Household Income and the sum of Incomes) or a vector of univariate variables (such as the combination of Age and Marital Status). In INSPECTOR's terminology univariate variables are called **atomic variables**, while vectors of variables are called **composite variables**.

Certain atomic variables exist regardless of validation; they are those variables on which data are collected in a survey, like Income. These atomic variables are called **actual variables**. Other atomic variables are defined due to the existence of a validation rule, like the sum of Incomes and its difference from Total Household Income, in which case they are called **implicit variables**. Composite variables are always defined due to the existence of a validation rule.

Just in order to recapitulate, we have three kinds of variables:

- **actual variables**, which are the atomic variables constituting the primary dataset,
- **implicit variables**, which are atomic variables not in the primary dataset but required for the application of a set of validation rules, and
- **composite variables**, which are vectors of atomic variables (actual and/or implicit) defined because of some validation rule.

Some implicit or composite variables require intermediate variables for their definition. These are also implicit variables. For example, *sum of Income* in the example validation rule mentioned before is an implicit variable leading to the final atomic variable Total Household Income – sum of Income.

Data validation is applied by checking whether each variable takes a value inside its domain. The domain of an atomic variable will be either a range or a union of ranges of values if the variable is continuous or a set of values if the variable is categorical. In each case the domain may be the union of a number of **domain elements**. For example, if a variable takes values from the codes of a classification and values from different levels of the classification are allowed then each level is a domain element. Another example are numeric variables whose domain is a large range of values with

the exception of a smaller set of values. Then we can define two domain elements, the range and the set of excluded values. For composite variables the domain is a subset of the Cartesian product of the domains of its components.

All implicit and composite variables must be physically created and must belong to a record type. According to the INSPECTOR approach, they must be part of the base record type of the validation rule that 'defines' them. Therefore the records of the primary dataset are augmented by the addition of all necessary implicit and composite variables. A copy of the primary dataset is made and its records are augmented with the addition of all these variables, resulting in a dataset which we call the **enhanced dataset**. Validation is then applied to the enhanced dataset and its results are related back to the primary one.

Implicit variables are of several kinds. The kind of a variable depends on its origin and on the way it is constructed. The definitions and examples below will make this clearer:

- **Secondary variables:** these are atomic variables which take their value from datasets other than the primary one. For example, if in the Census dataset we have the Income of each person and there is a validation rule comparing it to the income declared by the same person to the Revenue Office, then we define, in record type Person, a secondary variable called TaxIncome, which takes its value from the Revenue Office's databases. Similarly, secondary variables may take their values from datasets of past instances of a survey. The datasets which provide the values of secondary variables are called **secondary datasets**.
- **Derived variables:** these are variables which take their value from other records of the same dataset, than the record they belong to. They are of three sub-kinds:
 - **Hierarchical:** these derived variables take their value from records of the same observation but higher in the hierarchy of record types or from the metadata record.
Suppose for example that the Dwelling record of each observation contains variable Number of Rooms, and that in each household record there is variable Rooms at the disposal of the household. A logical validation rule requires the latter variable to be at most equal to the former. Its base record type is Household and in order to implement it we need a hierarchical implicit variable, Dwelling Rooms, which will be a copy of Number of Rooms, stored in each household record and taking its value from the dwelling record of the same observation. Dwelling Rooms is required in order to lead to the atomic variable Dwelling Rooms - Rooms at the disposal of the household which implements the rule by having [0,...) as its domain.
 - **Referential:** these variables take their value from a variable in any other record (of the same or of different record type), in any observation, where the record is identified by the value of a reference key.
For example, suppose that in the Census dataset each person receives an ID (within the household) and that each person declares his/her age in years and the ID number of his/her mother if she is in the same household. Suppose also that there is a validation rule specifying that a

mother must be at least twelve years older than each of her children. We then need, in each person's record, a referential variable called Mother's Age. The record of the mother (from which Mother's Age will take its value) is identified by matching variable Mother's ID of the current record with variable ID of all other person records in the household; ID is the reference key. Mother's Age is required so that we can form atomic variable Mother's Age - Age which implements the rule by having [12,...) as its domain.

- **Iterative:** these derived variables take their values from the records of the same record type and of the same observation, which precede or succeed the current record according to some predefined sequence. Suppose that in the Census dataset we need to check whether person records were 'lost' during data entry. If in a household all records have been stored then the persons' ID values should be successive numbers from 1 up to the size of the household. To implement this validation rule we create in each person record (except the first in each household) an iterative derived variable, Previous record's ID, which takes its value from the ID of the preceding record in the dataset. The rule is implemented through the atomic variable ID - Previous record's ID which has {1} as its domain.
- **Computed variables:** these are obviously variables computed as functions of other variables. Examples have already been given above: Dwelling Rooms - Rooms at the disposal of the household, Mother's Age - Age, ID - Previous record's ID. The functions may be of several kinds:
 - **Mathematical:** this category includes all mathematical functions (summation, subtraction, exponentiation, etc.), that may be applied on a number of variables which are all in the same record.
 - **Range:** such functions transform a continuous variable into a categorical one by assigning codes to its values according to the range its value lies in (e.g. code 1 for values ≤ 1000 , code 2 for values > 1000).
 - **Look-up:** these functions recode categorical variables by assigning new codes to their old codes; grouping of old codes is also possible by assigning the same new code to several old codes.
 - **Aggregation:** this category includes functions that apply on multiple instances of the same atomic variable, such as *mean*, *standard deviation*, *sum*, etc.
 - **Array-set:** this category includes functions such as *count*, *min*, *max*, *at least one*, which traverse a set of values and make calculations, excluding aggregation functions.

Therefore, with the use of the appropriate implicit and composite variables, all rules become within-record rules.

The structure of the enhanced dataset is described in a **dataset template**. The template contains the following information:

- The definitions of the record types and of their hierarchy: these will be supersets of the record types of the primary dataset because a number of implicit and composite variables will have been added to them. The definitions will contain a declaration of all the variables that comprise each record type.

- For each hierarchical, referential or iterative implicit variable, the definition of the record it comes from and the variable it copies its value from.
- For each computed variable, its formula and arguments.
- For each composite variable the names of its components.
- **Precondition variables:** these are variables which, if they fail a validation rule, do not allow any more validation dependent on it to be applied on the same observation. For example, if a variable records the Status of a dwelling in the Census then if this variable takes value ‘Empty’ no more validation needs be applied since no more data will be available. This is a precondition variable.
- The domain of each variable.

It is obvious that there is a direct relationship between a set of validation rules and a dataset template. If a rule is relaxed certain variables will possibly be removed from the template; correspondingly, if a rule is added, the template may be enriched with new variables.

A validation system needs information about the source of each actual and secondary variable. This information is provided with the definition of **abstract sources**, which are generic names representing the datasets providing the variables (e.g. ‘Primary’, ‘Current Tax File’, ‘Last Labour Survey File’, etc). The structure of each abstract source (record types and variables) must also be provided but in a less detailed form than for the dataset template; for secondary sources the only information that needs to be provided for each record type are record identification variables, the variables that will be imported in the enhanced dataset and the variables whose aggregates or array-set function results will be imported. Moreover, domains do not have to be provided. This lighter structure is called a **source template**.

The combination of dataset template and source templates² refers to a particular survey and a particular set of validation rules but not to a particular instance of the survey. In other words, if the structure of the survey does not change, the same template will be applied in future instances. In order to achieve this, the template defines the variables with abstract names (e.g. ‘Income’, ‘Education’, etc.). When the template is to be applied in a particular instance of a survey, a **validation project** is defined in which abstract source names are mapped to actual filenames and abstract actual and secondary variable names are mapped to actual field names in the files. Of course, if the structure of one of the files changes then at least the respective source template will have to be modified accordingly.

3 THE INSPECTOR SOFTWARE PROTOTYPE

In order to test the INSPECTOR approach in practice, the project has developed a software prototype which implements it. This prototype has been used in real life examples, as shown in section 4.

In the current section we present this prototype. We avoid technical details, which may be found in project deliverables. Our presentation overview the functionality offered by the prototype, its interface and its design.

² For brevity reasons one may call the whole combination a dataset template.

The software is available for use via the Internet. Parties interested in obtaining access to it are invited to contact the authors.

3.1 FUNCTIONALITY

The processes carried out with the help of the software can be summarised in two main groups, namely “dataset template management” and “validation project management”.

The main goal of dataset template management is the declaration and maintenance of validation rules, while the main goal of validation project management is the actual validation of data. These two groups can be logically divided into tasks.

Dataset template management tasks:

- a. Declaration of the Template structure (Template, Record Types, Variables etc).
- b. Declaration of the domain for each Variable of the template (this part is actually the declaration of validation rules according to domain-based validation).
- c. Declaration of the Source Template structure (Source Template, Source Record Types, Source Variables etc).
- d. Declaration of the matching between Source Variables and Actual variables.

Validation project management tasks:

- a. Project Creation. (Project metadata, Template selection, temporary database schema selection for storage of enhanced dataset).
- b. Data loading (provision of source XML file(s) to the system, uploading of raw data from them).
- c. Data enhancement (generation of data values for all implicit variables required for the validation process).
- d. Data validation (the actual validation of the dataset).
- e. Reporting (validation results matrix, metrics, exporting/importing of enhanced datasets).

All of these tasks, as well as some subsidiary ones, are performed by the software in a uniform manner through the supplied user interface.

Templates and whole validation projects may be shared between users. More specifically, the tool enables users to connect to and copy templates to/from template repositories which are physically located in remote computers (in an organisational intranet environment or via the Internet).

It should be noted that the prototype currently can upload raw data only in XML form or in ASCII format with a particular structure. However, the tool can be extended with the development of converters between other formats and XML.

3.2 USER INTERFACE

The main component of the interface, from which most of the user actions are initiated, is the tree-like structure in the left pane of the user’s screen. This tree

structure is considered to be the preferred one for presenting, in a user-intuitive way, information that somehow entails a hierarchical structure, which is the case for the Template structure.

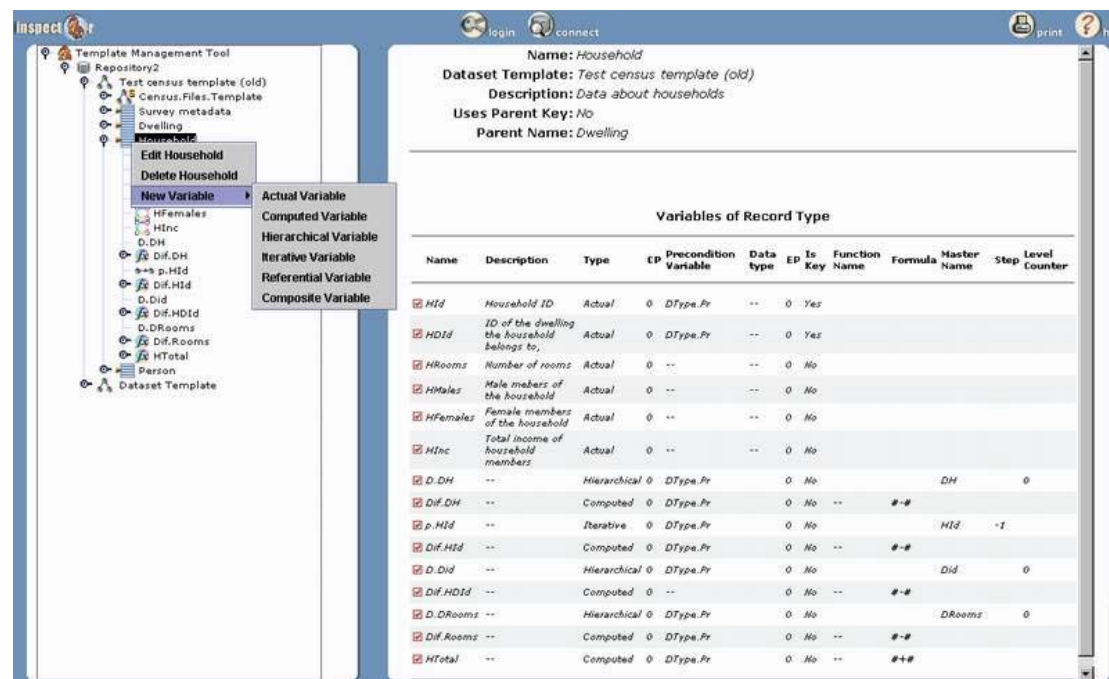


Figure 1: A view of the interface of the INSPECTOR software prototype.

The right pane of the user's screen is where information regarding the selected tree item is displayed for declaration, modification and viewing purposes. Also whenever user feedback is required, this is done through forms displayed in the right pane. Figure 1 presents a view of the interface.

The way that a user may act on a tree item is by selecting it (the tree item becomes highlighted and its detailed information is displayed on the right pane) and then by right clicking anywhere on the tree pane. A pop-up menu appears which offers to the user the actions which apply to the selected tree item.

There are three basic actions that a user may perform on a selected tree item:

- a. Edit the item.
- b. Delete the item.
- c. Create a (hierarchical) child of this item.

For example, by right clicking on a Record Type item, a pop-up menu appears with the command choices: 'Edit RecordType', 'Delete RecordType' and 'New Variable' (which is considered as a child of a RecordType).

In some cases, a sub-menu may appear next to the main pop-up menu offering the user more specific choices. In the example above, when the user points with the mouse over the 'New Variable' choice, a sub menu appears offering the choices 'Actual Variable', 'Computed Variable', 'Hierarchical Variable', 'Iterative Variable', 'Referential Variable' and 'Composite Variable'.

It is also possible that a tree item may have more than one type of child items (in terms of their place in the tree hierarchy) e.g. a Dataset Template has two child-types, Record Type and Source Template. In such a case, when the user right clicks on a selected Dataset Template, he/she is presented (besides the edit and delete commands) with both the 'New RecordType and 'New SourceTemplate' commands.

Besides the three basic actions, more actions may be applicable to an item depending on its type. For example, on a Dataset item the menu command 'Load Dataset' is also displayed.

3.3 DESIGN OVERVIEW

The high level architecture of the INSPECTOR System is depicted in Figure 2.

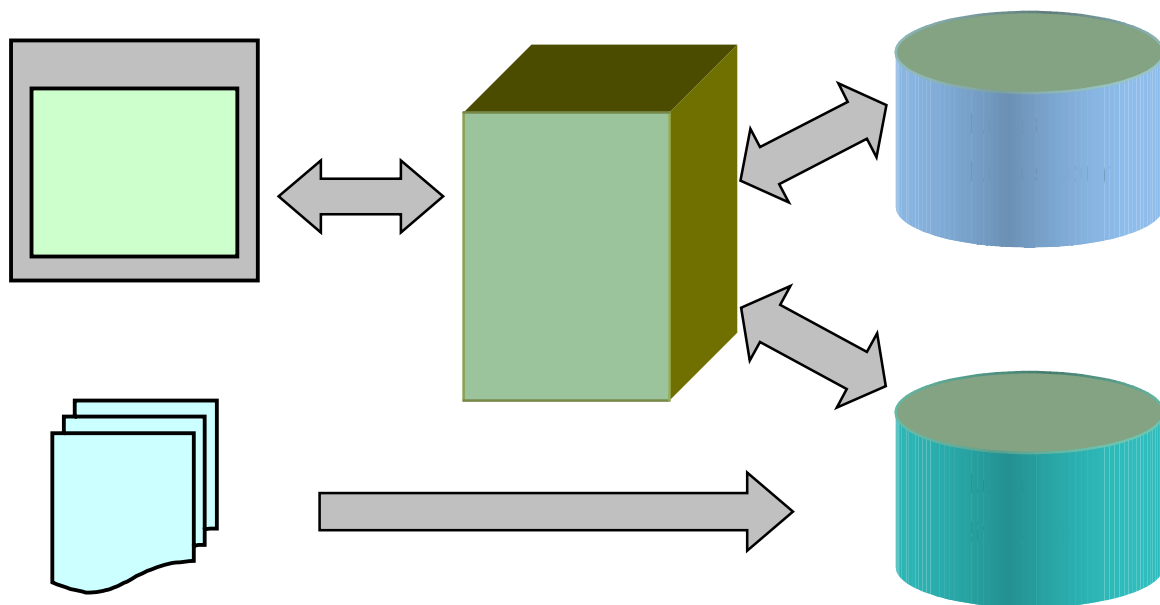


Figure 2: INSPECTOR System - High Level Architecture.

The System consists of three distinct layers, the Database Layer [DBL], the Application Layer [AL] and the User Layer [UL].

The DBL comprises two database parts the Rules Repository [RR] database and the Local Storage [LS] database. The RR is where the rules are stored, while the purpose of the LS is storing user's data i.e. datasets. The two databases reside in a relational database management system (RDBMS), which can, but not necessarily, be common for both.

The AL, which is depicted in the diagram by the 'Application Server' icon, 'lies' between the user and the databases and is the main software component of the INSPECTOR System. It is a Web Application and runs on an Application Server.

The UL is a thin client consisting of a Java enabled web browser that runs on the user's machine.

Note: the ‘Data Files’ are not part of the INSPECTOR System but rather of its environment.

Although INSPECTOR is essentially a web application, it is possible that all of its parts reside on a single machine. Thus the System can support the needs of large corporations and organizations, installed on a corporate intranet (even providing Internet-based access), as well as the needs of a single statistical researcher, installed on his/her machine.

4 PRACTICAL APPLICATION – INE’S EXPERIENCE

4.1 INTRODUCTION

In this section the authors will try to overview the most relevant and objective aspects of the practical use of the INSPECTOR system, based on the experience got during the Pilot evaluation phase of INSPECTOR project.

Some trial tests were made, in order to measure the applicability and capabilities of the prototype. The tryouts were meant to emulate as much as possible “real production” conditions.

The most crucial, though obvious, objective for this trial was to test the features available in the INSPECTOR system and to check their conformity to the original set of requirements.

The survey choice was a key factor for the feasibility of the evaluation and was performed under the following premises:

1. Capability to test all the features (the more demanding the better) made available by INSPECTOR;
2. Comparability of results;
3. Applicability of the software to new problems, through the application of already existing templates.

4.2 ENVIRONMENT

4.2.1 General

The general INSPECTOR system architecture is based on a thin-client multi-tier model. The base architecture may be implemented to serve on an Internet, Intranet or both environments.

At Instituto Nacional de Estadística (INE henceforth), we successfully tested both Intranet and Internet installation environments. The second showed to be much more valuable since it allowed for Template Replication between Repositories and for a close monitoring and interactive support from the software team.

Inter-Repository functionalities were tested demonstrating the system’s ability to be used as a Template distribution tool.

The UTF-8 character set was used for internal representation allowing for a further detailed inter-regional (pan-European) integration application.

4.2.2 Human Resources

In order to implement the surveys' templates and projects for INSPECTOR Pilot, a team was constituted. Inspector implementation required over the Pilot a team that ranged from 4 to 6 elements distributed in the following functions:

- The team manager: Key element in the implementation of templates into the system. Apart from planning and coordinating operations the team manager constituted a link between Statistics and Survey experts and the system.
- IT experts: Team that dealt with the installation and update of the software and support systems.
- Statistics experts: Key elements during the Pilot. Each expert is responsible for the trial referring to his / her survey and has the function of implementing it along with the validation rules.

4.2.3 Chosen cases

Surveys were chosen in order to pursue several aims, the most important of which was to guarantee that the largest number of identifiable types of validation rules were checked. It was also aimed to test as many features of the INSPECTOR software as possible.

The second aim in the process of choosing surveys was to pick the most representative surveys of National Statistical Institutes (NSIs henceforth). Finally, the third aim consisted in conveying different types of surveys, in what concerns periodicity, sampling processes and population targets.

During the first stages of INSPECTOR project a number of surveys were analysed in terms of their validation needs. For these surveys a set of validation cases and rules, were defined and of course, their individual characteristics were collected and analysed.

Then, it was natural to select the surveys for trial among this set of analysed data flows. In first place, a very clear idea of each one of the surveys was formed and its individual characteristics already known. Secondly, since the identification of the validation cases was already done, this would reduce the time needed for preparing the data for the system and consequently the required time for the overall pilot process. Another selection criterion was the easy access to the actual data set. Real data availability should be assured for the testing.

With this in mind, the surveys chosen to be used for INSPECTOR evaluation were the Labour Force Survey (LFS) (mainly a longitudinal coherence study) and the Turnover Index Survey (IVNEI - an STS survey).

Some specific features were tested in each selected survey.

LFS – one of the most important surveys of NSIs. It is essential as it provides vital information for Governments. It is also essential for Eurostat to have a general idea of the quality level of data and statistics therefore computed. It is a quarterly survey (in the Portuguese case), with a long list of variables. In our case, a great part of the variables was dropped since they were not needed for the testing purposes. Structural quality was the focus along with Time and record consistency that were the main features to be tested. This way, around 70 actual (picked from the source data) variables are used. Around 25 validation rules were defined and inserted in the system, which resulted on about 140 non-actual (derived) variables created. About 125, 500, 5000, 16000 and 32000 statistical units were used during the testing phase. The time needed for building up this test structure (preparing the Template) was about two weeks.

IVNEI – one of the most important exhaustive surveys for NSIs and a member of a set of about 20 short-term statistics (STS) surveys. In theory, all companies are interviewed. The response rate is high. There's a minimal quality check system already implemented for this survey. This would allow us to evaluate the performance of INSPECTOR. The whole set of actual variables was introduced in the system (37, which includes variables from periods different from the one being tested). Also, around 26 validation rules were introduced, which implied the creation of around 25 derived variables. The number of statistical units employed on the tests was about 220, 554, 1110 and 2205 for each test. Time of execution was about one week.

4.3 THE CAPABILITY TESTS

4.3.1 The abstraction application

The effective trial revealed, in practice, what was already envisaged in prior phases of the project: the “rule oriented” validation rules translation into “domain oriented” variables and domains is a complex and tiresome task when performed manually.

Validation experts are not used to the “domain oriented” abstraction and it was found that the learning period is long until some minimal proficiency is achieved.

As far as the simple rules (low complexity; simple checks) are concerned, the domain-oriented approach was found quite straightforward to implement. Inversely, when the rules shows a complex degree of logical dependencies, the translation requires a lot of human resources and even logic expertise to get the task done.

The time spent on translating and fine tuning the validation rules demonstrated the above during trial: 40 man.day to translate about 25 complex logic validation rules into about 200 variable domain checks (including all tuning and corrections).

4.3.2 Rule Translations

The extraordinary difference on coding “rule oriented” (RO) validation rules and the functionally equivalent “domain oriented” (DO) ones is quite well illustrated by the following table:

Table 1: Relations between RO and DO

| Type | # | (Note) | IVNEI | LFS |
|--------------------------|-----------|--------|-------|-----|
| Rule Oriented | Rules | 1 | 26 | 26 |
| | Variables | 2 | 37 | 75 |
| Domain Oriented | Rules | 3 | 34 | 154 |
| | Variables | 4 | 62 | 204 |
| File Types | | | 5 | 2 |
| Levels of data structure | | | 1 | 3 |

Notes:

- | | |
|---|--|
| 1 | Number of Logical based rules |
| 2 | Total number of variables involved (taken on the actual variables) |
| 3 | Total number of variables with defined domains |
| 4 | Total number of variables involved |

4.3.3 Suitability

It was easy to find out that the system is most suitable to be used on a “one template-many files” basis than the reciprocal. Indeed, it performed extremely well on the execution phase of the project, when the templates already defined (the validation rules) were applied against several files, or even transferred between repositories to be applied on a distributed way. Inversely, the exercise of applying successive different sets of validation rules to the same data, showed to be somehow restricted to the relaxation of certain rules (domain checks) becoming a cumbersome task the tuning and rewriting of entire alternative validation rule sets.

Nevertheless, the distributed aspect of the system architecture, and the ease of the template dissemination, recommends it for use on environments with such requirements (i.e. Eurostat data collection systems).

4.3.4 Some examples of validation rules translation:

The following examples, were taken from the Pilot execution at INE, demonstrate quite well the degree of difficulty (or simplicity) associated with RO validation rule translation into DO domain checks.

1. Easy Variable (in terms of RO approach)

This case reveals the simplicity of the representation of the basic domain check validation rule.

Sex: check if all cases have this field reported; check if it is 1 or 2. (*0% LFS recorded error rate*)

RO: sex in { 1,2 } ; nulls not allowed;

DO: sex is an Actual variable; Domain of Sex is Categorical; Elements of the domain are codes 1 and 2.

2. Medium Difficulty Variable

This case demonstrates that a medium difficulty RO validation rule can become a very extensive and complex cascade of DO checks.

T_Compdate12: check consistency of dates in answers given by the same person in different interviews (different data sets). This variable is computed using 2 actual variables but it is preconditioned by a third variable. The third variable is a cascade of relationships among variables. The following graph elucidates it. In the end, some characteristics of each variable are given. (~0.07% LFS recorded error rate)

RO: Check if the date of first job is the same over time, independently of the employment status; or : “if people go from unemployed status to employed status, then Data_Fi_Job2 (t0) = Date_Fi_Job (t1); or even: IF ([unemployed(t-1)=1] AND [employed(t)=1]) THEN Data_Fi_Job2(t-1)=Data_Fi_Job(t);

DO: T_Compdate12 is a Computed Variable; with Formula= #-# ; Arguments are Data_Fi_Job2 and Data_Fi_Job; Domain of T_Compdate12 is Numeric range, Elements of range are 0 and 0; with interval type “[]” (representing [0:0]). The variable is preconditioned on the result of Unemp_Emp.

Where:

- Date_Fi_Job2_p – date of the start of the first job ever (unemployed person)
- Date_Fi_Job – date of the start of the first job (employed person)

Being preconditioned by:

- Unemp_emp (precondition derived variable check) – which represents the change of employment state between interviews (un->employed) (this domain check may report error not to be accounted on the final data quality metrics).

DO: Unemp_emp is a Computed Variable; with Formula= #+# ; Arguments are Unemployed_p_logical and Employed_logical; Domain of Unemp_emp is Numeric range, Elements of range are 2 and 2; with interval type “[]” (representing [2:2]).

Where:

- Unemployed_p_logical – represents the prior unemployment state
- Employed_logical – represents the actual employed state,

Both derived from the requirements to unemployed stated by ILO classification;

Being computed as:

- Unemployed_p_logical decode 6->1; 0

DO: Unemployed_p_logical is a Computed Variable; with Formula= DECODE(#,6,1,0); Argument is Unemployed_p_forlogical; no Domain defined.

Where:

DO: Unemployed_p_forlogical is a Computed Variable; with Formula= (#+#+#+#+#+#+#+); Arguments are work1_p_logical, work2_p_logical,

work3_p_logical, availability_p_logical, seek_work_p_logical,
age_shadow_p_logical; no Domain defined.

The Employed_logical Computed variable is defined in a similar manner.

Every variable is of the “Computed” type, except for the *basic* variables, which are obviously “Actual”. The “Actual” variables are: Date_Fi_Job2_p; Date_Fi_Job; work1_p; work2_p; work3_p; availability_p; seek_work_p; age; work1; work2; work3.

3. *Difficult (Extreme) Variable*

This case demonstrates that an apparently complex RO case can become a very simple domain definition.

T_HHIN_Struct_error: checks the structural files coherence, when there is, for instance, different record types, some of them being parents of others. Records on a certain level of the hierarchy indicate the number of its child records and that value must be checked against the actual number of child records. A 3,65% error rate was found on the HH data set of LFS.

RO: Check if the reported number of persons of the HH (household) is equal to the count of the corresponding Person records.

DO: T_HHIN_Struct_error is a Computed variable; with Formula=(#-#); Arguments are Count_Ind_I and N_Individuals; Domain of T_HHIN_Struct_error is Numeric range, Elements of range are 0 and 0; with interval type “[]” (representing [0:0]).

Where:

- Count_Ind_I (counts the cases in the level below using function DET_COUNT) based on different Individual_Ids; and
- N_Individuals (pertains to HH record type and is supposed to be equal to the count)

All variables are computed, except the *actual* Individual_Id and N_Individuals, the former pertaining to the Individuals Record Type and the latter to the Household Record Type.

4.3.5 Data structures

INSPECTOR software allows for hierarchical structures to be defined both at the data level as for the classification levels. This characteristic is inherent to the system object model, which represents the most common structures found on the validation cases analysis performed at the early phases of INSPECTOR project. The vast majority of the data structures found in NSI are of the hierarchical kind.

During the Pilot two types of structures were tested; a flat structure (IVNEI) and a 3 level, 3 entity simple hierarchical structure. It was found that the system managed quite well both kinds of structure and in a flexible way. Indeed both cases obliges to

the use of data pertaining to different data collection periods, hence, to load and manage two similar structures representing the respective data instances over time.

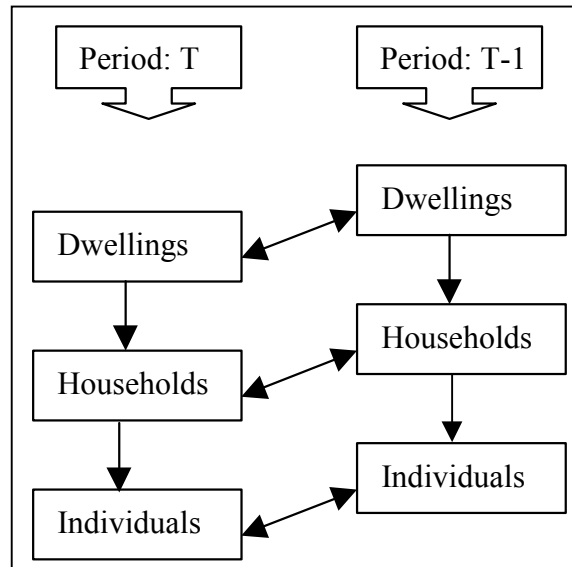


Figure 3: LFS Study Data Structure.

Data Loading in INSPECTOR is a straightforward task. Tests were performed by loading data contained on XML files and also in pure comma separated ASCII files. To allow for this last functionality, the system presents the user with the possibility of mapping the relative position of the attributes, present in file, with the Actual variables defined on the Data Source template, (which in turn are linked to the data set Template).

Classification loading was also straightforward. The interface is intuitive and the operation resumes to one step. However, the classification file had to follow an XML data schema, so one must account for preparation time. For INE trial the classification files were translated on courtesy of Agilis. We did not take measurements on loading time, however our impression is that it is negligible. The last (known) version of the software already allows to import classification files in ASCII format.

As an example of the loading performance pattern, follows a table containing the recorded LFS data sets loading time (the left column represents a label).

Table 2: LFS - Record Loading Times

| LFS-Study | | | | |
|-----------|-------------|-------|-----------|---------|
| Dataset | Records | | Loading | |
| | Individuals | Total | Time[Sec] | Rec/Sec |
| 125 | 62 | 120 | 5 | 24 |
| 512 | 275 | 552 | 15 | 36.8 |
| 5000 | 2916 | 5973 | 115 | 51.9 |
| 16000 | 9266 | 18794 | 360 | 52.2 |
| 32000 | 17666 | 18259 | (lost) | (lost) |

(*) – The elapsed time measurements were not enough accurate to be considered

4.3.6 The process

The simplified process diagram for the use of INSPECTOR system follows.

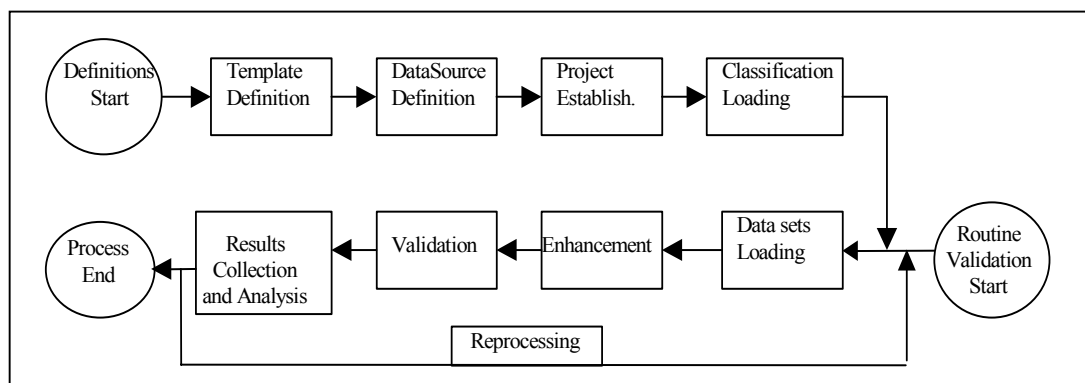


Figure 4: Validation Process with INSPECTOR.

As can be observed, the repetitive use of a Template definition (Validation rules) is preponderant over the repetitive definition of Templates. This particular feature allows to classify INSPECTOR as a tool dedicated to routine validations based on predefined Templates, which is considered as a harmonisation and dissemination oriented system.

4.3.7 The results

INSPECTOR presents the results of validation in the form of data quality metrics tables. The metrics are derived from counting for each domain check failure, and organised by Variable (say, by Rule) or by Record.

This kind of metrics is useful for the overall evaluation of the data quality. The system complements the reports with a “drill-down” capability, that lets users observe clearly the faulty items by showing the records and variables with failed domain checks.

Indeed, this data, is collected during the validation process, refers to every defined variable that failed a test, including the precondition ones. This detail is being subject to a reworking process of the application, since some variables (preconditions and other implicit auxiliary variables - even if they have to be checked against a domain) should not account for the overall quality score.

Some examples of the main produced outputs follow.

An example of the Quality metrics summary table:

Table 3: Example - "Validation Set Report"

Rules error rates for each record type in validation set vs_0125 of dataset ds_0125 from project LFS run v1 with template LFS run v1

| Record type name | Rules errors rates for the total number of records | | | | | | | | Error rates for the failed records | | | | | | |
|------------------|--|-------|------|------|------|------|--------|--------|------------------------------------|------|------|------|------|--------|--------|
| | at least 1 | 1 | 2 | 3 | 4 | 5 | > 5 | > 10 | 1 | 2 | 3 | 4 | 5 | > 5 | > 10 |
| Dwelling | 17.65 | 17.65 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 100.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| Household | 24.14 | 24.14 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 100.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| Individuals | 100.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 100.00 | 100.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 100.00 | 100.00 |

Failures “by Record”

For each record type a table is generated pointing out the percentage of failure. The summary table indicates the percentage of failure for each record type.

Table 4: Summary Table - "View Failed Records"

Validation Set "vs_0125"

Failure for each record Type

Click to see failed Records

| Record Type | Records failed (%) |
|-----------------------------|--------------------|
| Dwelling | 17.65 |
| Household | 24.14 |
| Individuals | 100.00 |

Detail from “View failed records”:

Table 5: Detail Tables - "View Failed Records"

Validation Set "vs_0125"

Records of Record Type "Dwelling" that failed

| Record order | Id_DW | Failure (%) |
|--------------------|-------|-------------|
| 31 | 22132 | 25.00 |
| 32 | 22133 | 25.00 |
| 33 | 22137 | 25.00 |
| 14 | 12118 | 25.00 |
| 15 | 12119 | 25.00 |
| 16 | 12123 | 25.00 |

Failures “by Rule”

For each record type a table is generated pointing out the percentage of failure by rule. A summary table indicates the percentage of failure for each record type.

Table 6: Summary Table - "View Failed Rules"

Validation Set "vs_0125"
Failure for each record Type
Click to see failed Rules

| Record Type | Records failed (%) |
|-------------|--------------------|
| Dwelling | 17.65 |
| Household | 24.14 |
| Individuals | 100.00 |

Detail from "View failed Rules":

Table 7: Detail Tables - "View Failed Rules"

Validation Set "vs_0125"

Rules failure for Record Type "Dwelling"

| Variables | Failure (%) |
|-----------|-------------|
| ID_P | 17.65 |

Validation Set "vs_0125"

Rules failure for Record Type "Household"

| Variables | Failure (%) |
|---------------------|-------------|
| ID_P | 17.24 |
| T_HHIN_Struct_error | 6.90 |

Exports

INSPECTOR systems allows, in addition, to export the overall error detection matrix (record vs. failed domains) which can be very useful for the automation of the subsequent processes (imputation, for instance).

Correctness of Results

During the evaluation phase, it was shown that apart from the misleading indicators that result from the precondition and other non-relevant implicit variable failure accounting, the system performed the error detection very precisely. Sometimes some errors were found to be induced by incorrect RO validation rule translations, which reinforces the users' conviction about the system accuracy and correctness.

Performance measures

An example of the performance marks obtained by INSPECTOR processing the LFS and IVNEI cases follows.

Table 8: IVNEI - Enhancement and Validation Measures

| IVNEI | | | | | | |
|-----------|---------|-------|------------|----------|-----------------|----------|
| Dataset | Records | | Time [sec] | | Rates [rec/sec] | |
| | A | Total | Enhance | Validate | Enhance | Validate |
| 0024 | 24 | 7013 | 6 | 5 | 1168.83 | 4.80 |
| 0220 | 197 | 7186 | 5 | 30 | 1437.20 | 6.57 |
| 0554 | 531 | 7520 | 23 | 75 | 326.96 | 7.08 |
| 1110 | 1076 | 8065 | 45 | 125 | 179.22 | 8.61 |
| full-2205 | 2216 | 9205 | 85 | 224 | 108.29 | 9.89 |

Table 9: LFS-Study - Enhancement and Validation Measures

| LFS-Study | | | | | | |
|-----------|-------------|-------|------------|----------|-----------------|----------|
| Dataset | Records | | Time [sec] | | Rates [rec/sec] | |
| | Individuals | Total | Enhance | Validate | Enhance | Validate |
| 125 | 62 | 120 | 5 | 20 | 24.00 | 3.10 |
| 512 | 275 | 552 | 7 | 70 | 78.86 | 3.93 |
| 5000 | 2916 | 5973 | 37 | 720 | 161.43 | 4.05 |
| 16000 | 9266 | 18794 | 80 | 2 206 | 234.93 | 4.20 |
| 32000 | 17666 | 18259 | (lost) | 4 858 | (lost) | 3.64 |

Although the evolution of the curves seems to be steady, we have not been able to depict a consistent pattern between the tested cases, as it happened for the loading functionality. This may have been caused by a lot of factors not accounted when the measurements were taken, namely factors related to the architecture, database server load, etc. Another possible factor is the way historic data was treated on both cases. For IVNEI case, the only change between trials was the number of primary records (those to be tested), whereas the historic data was maintained at its full size. On the contrary, for the LFS case, the historic data was tailored to match as possible the size and sample rotation characteristics of the primary data.

Comparisons

Due to the lack of consistent error reporting data from real production process at NSIs a serious comparison of results and performance marks could not be performed. However correctness of INSPECTOR results was demonstrated, during the Pilot, making use of simulation tools.

4.4 PILOT TEST CONCLUSIONS

In a broad sense, the system implementation was considered as being halfway between a “proof-of-concept” and a “Production” system.

The users managed to perform and measure validation and analyse results. The correctness of the detection and reporting was confirmed, which may allow declaring the system as reliable and precise.

The users found that the human interface of the system and also the concept beyond the process sequence could be much improved, and suggested a set of further improvements to be taken into account in future versions.

A list of user appreciation's/suggestions follows:

- Interactivity must be greatly improved in INSPECTOR to make it “commercially” viable. A more production oriented and agile user interactivity is needed (maybe with some kind of wizards);
- Re-working of the missing values processing concepts is an important point;
- Some flexibility should be given to the results and quality metrics reporting;
- It is important to re-work data input sections to allow, also, for database connectivity (via ODBC or ADO for instance);
- Re-work the operations sequence giving more priority (and an upper level on object organisation) to the Data Source Templates to match production preferences.

Finally, based on the experience of the Pilot it can be said that the Domain Validation model demonstrated to be innovative and capable of being used as a tool in Official Statistics.

5 CONCLUSIONS

In this paper we have presented an alternative approach to data validation, the first stage of data editing, where data are checked for errors and unusual values. According to this approach, each validation rule (edit) is ‘translated’ into the definition of a variable with a domain of allowed values. Computationally, all variables implied by the rules are generated by suitable software, their values are computed from the data and the values are checked as to whether they fall in the defined domain. Therefore all validation rules, irrespective of their complexity, become domain checks.

This approach demands from survey statisticians to operate under a new conceptual framework. This framework may in fact be ‘hidden’ under the prevailing framework of *declarative rules* (‘IF-THEN-ELSE’ statements), but it is still awkward at first sight. However, after practical experience of a few weeks, the conceptual difficulties disappear.

On the other hand, it can be argued that the approach requires care in its practical application. The definition of the dataset template (i.e. the specification of the rules in domain-based form to the validation software) can become very time consuming in the case of surveys with large numbers of variables and rules. This may lead to errors, even by experienced users. This argument confuses the approach, with the facilities, offered by the software prototype to users, for declaration of the template. The user has to make the translation of rules into variables and to specify the results to the system. However, the validation software can be equipped with a ‘wizard’ which receives rule statements in a simple high-level language and translates them into dataset template components. The approach would still be the same but its implementation would be simpler. Since the current prototype demonstrates adequately the effectiveness of domain-based validation in computational terms, the wizard was not implemented, due to resource constraints of the project. However, the

authors have made research into defining the mentioned high-level language, with quite advanced results, which are available upon request.

Even at the current state of the software prototype however, the user burden is great only at the initial specification of a template. It is anticipated that, once defined, a template will be used, without changes, for several instances of the survey it was designed for. Moreover, if the template is prepared by a central authority (e.g. EUROSTAT) for a survey which will be conducted by several national offices, then the template can be used by all of them, greatly increasing harmonisation of validation. The validation systems of the national offices can be connected via the Internet with the central authority's system for continuous sharing of template updates.

A second point to be noted about the approach, already mentioned earlier, is that it does not cover the whole editing process. The problems of error localisation and subsequent imputation were from the outset beyond the scope of the INSPECTOR project. Explicit error localisation, according to the Fellegi-Holt theory (Fellegi and Holt, 1976) requires recourse to the declarative rules and generation of implicit rules. The notion of declarative rules contradicts the present approach. On the other hand, imputation methodologies, like the New Imputation Methodology (Bankier, 1999), which forego error localisation in the Fellegi-Holt sense, can easily be combined with our approach. The incorporation of such functionality in the INSPECTOR prototype or its connectivity with suitable software is conceivable.

The main motivation behind the investigation of the approach was to investigate its effectiveness in computational terms. The approach effectively treats validation rules as data and therefore is in accordance with the logic of database management systems. In terms of execution speed, as was shown by the practical application, the system seems to perform quite acceptably on enhancement and validation phases. However, the lack of production benchmarks does not allow us to thoroughly evaluate this issue.

As a general conclusion, we believe that the approach merits the attention of practicing statisticians in Official Statistics. Especially its ability to promote harmonisation of validation practices (in terms of the rules applied) makes worthwhile the consideration of its adoption by the European Statistical System.

ACKNOWLEDGEMENTS

The work presented in this paper has been carried out within the context of the INSPECTOR project. The publication of the paper forms part of the project's dissemination activities. The authors thank all the members of the INSPECTOR project consortium for their contributions.

The authors would like to thank the Instituto Nacional de Estatística - Portugal, for the provision of the questionnaires and the sample data sets from the IVNEI and LFS surveys and all pilot evaluation material produced within the INSPECTOR project's context.

REFERENCES

- Bankier, M. (1999). Experience with the New Imputation Methodology used in the 1996 Canadian Census with extensions for future censuses. In *United Nations-Economic Commission for Europe Wrk Sessn Statistical Data Editing, Rome, June 2nd-4th*, working paper 24.
- Bankier, M. (2003). Current and future applications of CANCEIS at Statistics Canada. In *United Nations-Economic Commission for Europe Wrk Sessn Statistical Data Editing, Madrid, October 20th-22nd*, working paper 18.
- Chen, B. C. (1998). Set covering algorithms in edit generation. *Statistical Research Report 98/06*. Statistical Research Division, US Bureau of the Census, Washington DC. (Available from <http://www.census.gov/srd/www/byyear.html>.)
- De Waal, T. (1996). CherryPi: a computer program for automatic edit and imputation. In *United Nations-Economic Commission for Europe Wrk Sessn Statistical Data Editing, Voorburg, November 4th-7th*.
- De Waal, T. (2003). Solving the error localization problem by means of vertex generation. *Survey Methodology*, **29**, 1, 71-79.
- De Waal, T. and Quere, R. (2003). A fast and simple algorithm for automatic editing of mixed data. *J. Offic. Statist.*, **19**, 4, 383-402.
- Draper, L. R. and Winkler, W. E. (1997). Balancing and ratio editing with the new SPEER system. *Statistical Research Report 1997/05*. Statistical Research Division, US Bureau of the Census, Washington DC. (Available from <http://www.census.gov/srd/www/byyear.html>.)
- Engstrom, P. and Granquist, L. (1999). Improving quality by modern editing. In *United Nations-Economic Commission for Europe Wrk Sessn Statistical Data Editing, Rome, June 2nd-4th*, working paper 23.
- Fellegi, I. P. and Holt, D. (1976). A systematic approach to automatic edit and imputation. *J. Am. Statist. Ass.*, **71**, 17-35.
- Filion, J. M. and Schopiu-Kratina, I. (1993). On the use of Cherinkova's algorithm for error localization, *Technical report*. Statistics Canada, Ottawa.
- Garfinkel, R. S., Kunnathur, A. S. and Liepins, G. E. (1986). Optimal imputation of erroneous data: categorical data, general edits. *Ops. Res.*, **34**, 744-751.
- Granquist, L. (1997). The new view on editing. *Int. Statist. Rev.*, **65**, 381-387.
- Kovar, J. G, MacMillan, J. H. and Whitridge, P. (1991). Overview and strategy for the generalized edit and imputation system. *Working paper BSMD 88-007E*. Methodology branch, Statistics Canada, Ottawa.

Nordbotten, S. (1999). Strategies for improving statistical quality. In *United Nations-Economic Commission for Europe Wrk Sessn Statistical Data Editing, Rome, June 2nd-4th*, working paper 4.

Quere, R. (2000). Automatic editing of numerical data. *Technical report*, Statistics Netherlands.

Sande, G. (1979). Numerical edit and imputation. *Proc. 42nd Sessn Int. Statist. Inst.*

Schopiu-Kratina, I. and Kovar, J. G. (1989). Use of Chernikova's algorithm in the Generalized Edit and Imputation System. *Working paper BSMD 89-001E*. Methodology branch, Statistics Canada, Ottawa.

Todaro, T. A. (1999). Overview and evaluation of the AGGIES automated edit and imputation system. In *United Nations-Economic Commission for Europe Wrk Sessn Statistical Data Editing, Rome, June 2nd-4th*, working paper 19.

Winkler, W. E. (1998). Set covering and editing discrete data. *Statistical Research Report 98/01*. Statistical Research Division, US Bureau of the Census, Washington DC.
(Available from <http://www.census.gov/srd/www/byyear.html>.)

Winkler, W. E. and Chen, B. C. (2002). Extending the Fellegi – Holt model of statistical data editing. *Statistical Research Report 2002/02*. Statistical Research Division, US Bureau of the Census, Washington DC.
(Available from <http://www.census.gov/srd/www/byyear.html>.)

Winkler, W. E. and Petkunas, T. F. (1996). The DISCRETE edit system. *Statistical Research Report 1996/03*. Statistical Research Division, US Bureau of the Census, Washington DC.
(Available from <http://www.census.gov/srd/www/byyear.html>.)