DE GRUYTER
OPEN

Journal of Official Statistics, Vol. 34, No. 4, 2018, pp. 811–833, http://dx.doi.org/10.2478/JOS-2018-0041

Data Organisation and Process Design Based on Functional Modularity for a Standard Production Process

David Salgado[1], *M. Elisa Esteban*[1], *Maria Novás*[1], *Soledad Saldaña*[1], *and Luis Sanguiao*[1]

We propose to use the principles of functional modularity to cope with the essential complexity of statistical production processes. Moving up in the direction of international statistical production standards (GSBPM and GSIM), data organisation and process design under a combination of object-oriented and functional computing paradigms are proposed. The former comprises a standardised key-value pair abstract data model where keys are constructed by means of the structural statistical metadata of the production system. The latter makes extensive use of the principles of functional modularity (modularity, data abstraction, hierarchy, and layering) to design production steps. We provide a proof of concept focusing on an optimisation approach to selective editing applied to real survey data in standard production conditions at the Spanish National Statistics Institute. Several R packages have been prototyped implementing these ideas. We also share diverse aspects arising from the practicalities of the implementation.

Key words: Production architecture; key-value pair data model; standardisation functional modularity; process design.

1. Introduction

The modernisation and industrialisation of official statistical production has been at the centre of international and national activity in Official Statistics basically since the turn of the century, with the creation of the High-Level Group for the Modernisation of Official Statistics by the Bureau of the Conference of European Statisticians as a noticeable landmark (HLG-MOS 2017).

Indeed, this group was born with a clear strategic vision (HLG-MOS 2011) to streamline the statistical production by means of "different and better processes and methods tuned to delivering our products at minimal cost with greater flexibility and in cooperation between institutions" so that these "new and better products and services [are produced] more tuned to the way the world is operating today". Many outputs have been produced by the different groups operating under the umbrella of the HLG-MOS; these range from the establishment of diverse production standards (such as the Generic Statistical Business Process Model, GSBPM, the Generic Statistical Information Model, GSIM, the Common Statistical Production Architecture, CSPA, or the Generic Activity Model for Statistical

[1] Spanish National Statistics Institute, Paseo de la Castellana 183, 28046 Madrid, Spain. Emails: david.salgado.fernandez@ine.es, elisa.esteban.segurado@ine.es, maria.novas.filgueira@ine.es, soledad.saldana.diaz@ine.es, and luis.sanguiao.sande@ine.es

Organisations, GAMSO) over the promotion and development of streamlined statistical methods (e.g., UNECE 2017a) to capabilities and communication aspects (UNECE 2017b).

More recently, within the realm of the European Statistical System (ESS hereafter), the future of European Official Statistics is strategically envisaged by the ESS Vision 2020 (Eurostat 2014a) and its implementation portfolio in key projects, such as those focused on the European System of Business Registers (ESBRs), the Common EU Data Validation Policy (VALIDATION), the Shared Services for European Statistics (SERV), and the Digital Dissemination and Communication (DIGICOM), to name a few (Eurostat 2014b).

All these initiatives pose a challenge for statistical offices in their attempt to modernise their production, especially regarding the adoption of these new standards and practices: this is to be accomplished under the high pressure of product release calendars within the traditional stove-pipe production model and a decreasing amount of budgeted resources.

In this article, we want to present the ongoing efforts at the Spanish National Statistics Institute to bring a concrete plan for the modernisation of (a part of) the statistical production process into reality. Our rationale is that an official statistical production system constitutes a clear example of a human-generated complex system. We claim that to cope with this complexity, like with the design of computer systems, the principles of functional modularity are also of great value. These principles must fully integrate statistical production metadata, statistical methodology, and computer software design. These principles are often applied in the construction of software for the production of official statistics, but this is not enough. We claim that these principles must be applied *to fully integrate these three aspects of statistical production*, or else we would fail to cope with the complexity of the process. To illustrate our proposal, we show how we have developed a set of R packages to make a proof of concept that is already being applied in normal production conditions of several Short-Term Business Statistics (STS) at the Spanish National Statistics Institute.

Our proposal is based on two complementary elements. Firstly, for our data architecture, we make use of a key-value pair structure, in which keys are composed by making extensive use of the system of structural metadata. Secondly, adhering closely to the GSBPM and GSIM principles, for our statistical process architecture, we make use of the functional and object-oriented paradigms to incorporate modularity into the statistical methods. As we shall illustrate with the R packages, this paves the way for a natural posterior implementation in software tools. Our central message is thus *to bring modularity by design into the statistical process and the mathematical methodology itself and not just into the construction of computer tools*.

The article is organised as follows. In Section 2 we set up the generic approach, taking us from complexity as an essential trait of statistical production systems to the principles of functional modularity to cope with it. In Section 3 we argue that the international statistical production standards themselves implicitly suggest the use of a combination of the object-oriented and functional paradigms as a basis for building an information architecture. In Section 4, we detail the abstract data model that we propose to use as the central element of our proposed data organisation. Complementarily, in Section 5, we explain our proposed process design, and illustrate, with an example in statistical data editing, the application of modularity principles on a very concrete statistical methodological approach to selective editing. In Section 6, we share diverse aspects

regarding the implementation of this proposal, including the software tools development. We close with conclusions and future prospects in Section 7.

2. Generic Approach: From Complexity to Functional Modularity

The need for modernisation and industrialisation of official statistical production can be immediately argued from the very concept of *complex system*. The key features of a complex system are (Saltzer and Kaashoek 2009) (i) a large number of components, (ii) a large number of interconnections between these components, (iii) many irregularities in these interconnections, since the lack of regularity is the rule rather than the exception, (iv) a long description of the system and its related management (Kolmogorov complexity), and (v) a team of designers, implementers, and/or maintainers to handle the system. It is evident that an official statistical production system is a clear example of a human-generated complex system.

This conclusion can be illustrated and motivated with a simple description of the production of diverse statistical operations at a statistical office. Let us just consider the execution phases of the process. Data collection needs to be carried out in different data collection modes (CAPI, CATI, CAWI, EDI and others) on a number of statistical units (business units, households, or people), usually in the range of tens of thousands for each survey in a mid-sized country like Spain. This is multiplied by the number of variables (data or metadata) associated with each unit. These data must be entered into the system, edited, treated, validated, and curated to produce the corresponding microdata sets. They are further processed to produce the aggregated outputs using the appropriate statistical methods. They are then finally treated for disclosure control and, if necessary, for seasonality and calendar effects adjustment before the due dissemination. Each production step and data and metadata element in the process is interconnected to some other element. For example, a change in a parameter in a validation rule during collection will need to be followed by a post-capture data editing revision and adjusted aggregation procedure (e.g., in variance estimation). Indeed, the interconnections between all elements cannot be described according to a given regularity, thus making explicit the *water-bed effect*: a slight modification of a process step may lead to major consequences in another process step. Given the current setting of the statistical process at production offices, the description of how to produce the statistics for any given survey is not only necessarily long, showing the imbricate set of process steps, but also, hardly standardised. Members of the production staff of two different surveys who carry out the same tasks in the process can seldom be interchanged, despite common standard mathematical procedures underlying the estimation. Moreover, the number of actors in the process to be coordinated, not only for a given statistical operation, but also for the set of surveys conducted at an office (not to mention a whole national or European statistical system) is very high, which introduces evident management challenges.

In our view, the concept of official statistical production as the combination of statistics and complexity lies at the core of the need for the industrialisation of the statistical production process: not only do you need to use sound statistical methodology, you must also cope with this complexity for an efficient production process. Traditionally, in our view, official statistics have been produced in an artisan way, in which each survey was

independently designed and executed. Moreover, in extreme cases, not only have there been diverse (occasionally even incompatible) data and process architectures in different surveys in the same office, but different agents within the same survey have also made use of unconnected architectures, which has rendered management of the whole process virtually impossible. Up to the present, the stove-pipe production model has been extensively followed.

On a more quantitative footing, the inefficiency of this stove-pipe approach can also be justified by the complex nature of the production system itself. As a complex system, it is subjected to the square law of computation (Weinberg 2011) (see also Saltzer and Kaashoek 2009), which in our case can be expressed in terms of resources versus the number of requirements on the system.

A simplified description of how to detect and correct errors in a process step can illustrate and motivate this law. A process step is, basically, a collection of sequential and concurrent production tasks for accomplishing a given objective within the process. We can easily assume that the potential number of errors is proportional to the size of the production step (i.e., to the number of tasks) and that they can occur randomly throughout the step. In principle, in a nonmodular approach, an error is detected after executing the process step, which is then fixed. The process step is then executed again to detect new errors. If the time to find an error is assumed to be proportional to the execution time, the total amount of time to clean the process step will be proportional to the number of errors multiplied by the necessary cleaning time per error. However, the latter is proportional to the number of errors itself. Thus, the total amount of time will be quadratic to the number of errors. This argument shows how a naïve sequential approach to production becomes unmanageable due to the complexity of the system.

Under this square law, it is clear that increasing the number of requirements on the system (due to the incessant demands on Official Statistics, for example new legal regulations, more disaggregated information and so on) will produce a quadratic increase in the demand of resources, which is unattainable. Complexity must be coped with to face these challenges. The need for modernisation derives from the complexity of the global statistical production process.

The bottom line of our proposal is that we believe that the common principles of computer system design jointly known as *functional modularity* (Saltzer and Kaashoek 2009) are of great utility in designing and implementing an efficient official statistical production process. It is worth noting that functional modularity comprises four elements, namely modularity, data abstraction, hierarchy, and layering. These principles should be applied not only to the development of computer tools: *the process itself must be designed along these lines by conjugating statistical metadata, statistical methodology, and software design*.

Modularity is already at the very heart of production standards (such as the GSBPM – see next section), where the production chain is broken down into different subprocesses. However, modularity per se does not help us cope with complexity; we need data abstraction as this allows modules to be designed and implemented independently of each other, except for their interconnecting interface. Statistical processes must be designed independently of each other so that only initial inputs and final outputs uniquely enter into play in the chained execution of a given set of processes. The details of the execution of each subprocess must be transparent throughout the entire process.

Layering and hierarchy are principles applied to design and implement modules to minimise the number of interconnections among their components seeking optimal efficiency. In our proposal, these principles will be translated into organising both data and process architectures into four layers. A bottom layer for the statistical methodology (purely mathematical in many, but not all, cases); a second layer for the finest-grained production tasks upon which more complex activities can be composed (third layer). Finally, a top layer to orchestrate the whole process with these elements will complete the process design. We insist on the idea that this structure *must be applied to the statistical processes themselves, conjugating metadata, mathematics and software design*, not just to the construction of computer tools.

3. From Metadata to Architecture

The starting point for concretising our proposal into data organisation and process design is the interrelationship between the GSBPM and GSIM standards. The GSBPM is an international production standard modelling the statistical production chain in eight phases, each one divided in different production subprocesses. This standard focuses on production activities. Complementarily, the GSIM is another international production standard providing a model for the information objects in the production process. The inspiring interrelationship between the two standards is represented in Figure 1, already originally appearing both in the GSBPM (UNECE, 2013a) and in the GSIM (UNECE, 2013b).

There is also an implicit reference to this interrelationship that appears in the name of the GSBPM level-2 subprocesses (*Design collection*, *Test production system*, *Calculate aggregates* and so on) with the clear structure *action* + *information object*. If several transformations matching Figure 1 are concatenated, where the output of a step is the input of the next one, and if each transformation is associated to each input object, we have the conception of a statistical production process as a sequence of objects defined through their attributes (GSIM-like information objects) and transformed according to their methods (GSBPM-like production tasks).

Our proposal suggests a step forward in this direction by extensively using the principles of functional modularity to substantiate this general view of the combination of both GSBPM and GSIM. Note that these standards do not make any explicit mention of these principles, although their spirit is there. Similarly, in the international DDI standard (DDI 2018) a modular scheme for the successive transformations on both data and metadata sets is provided. Here, we also include these data and metadata under the same modular view.

To implement this dual data-process view under the principles of functional modularity, we firstly need to provide a data organisation scheme to deal with information objects in a standard way. Indeed, the proposed scheme must be valid for all kinds of statistics (social surveys, business statistics, statistics based on administrative registers, and so on). In the

Fig. 1. Interrelationship between GSBPM and GSIM standards (taken from UNECE013a).

next section we present an abstract data model based on key-value pairs in this sense. Indeed, we will define an object class for representing data in any kind of statistical data processing subprocess.

Complementarily, a process design scheme also needs to be provided. We understand that every "unit of statistical production information" is defined through a set of attributes (GSIM-like part) and a collection of statistical transformations (GSBPM-like part). In other words, they are *objects* (Booch et al. 2007). Furthermore, these objects can be thought of as constituting a sequence of transient transformations also combining data and metadata. This enables traceability and auditability of the whole process.

Indeed, this is extremely evocative of well-known computing models (van Roy and Haridi, 2004): the object-oriented and functional paradigms. The application of these paradigms makes each transformation depend only on its object input – it becomes stateless, that is, depending on no previous production step (state, in rigour). A cautious reader may immediately argue whether those steps involving (pseudo)random number generation arise as an exception to this stateless sequence of transient transformations. In full rigour, one can consider the random number generation seed as an internal state of the transformation. However, in the spirit of those statistical methods involving random simulation, we can accept that two processes providing *statistically* similar results can be considered identical under the data organisation and process design we defend here even despite numerical dissimilarities. This is a natural way of implementing referential transparency, that is, a property by which the procedure can be replaced with its corresponding value without changing the behaviour and the result of the whole process. As a consequence, executing a referentially transparent subprocess will always provide the same value for the same input arguments, irrespective of the rest of the process. This is the functional paradigm. As for the object-oriented paradigm, we concentrate on its advantages to model complex objects, and on its characteristics regarding transformations. Thus, transformations are conceived under the functional paradigm and objects are understood and modelled using the object-oriented paradigm.

However, we need to be more concrete about how to combine these paradigms in statistical processes. Let us focus on the recommendations of the METIS group elaborated by their informal task force on metadata flows (ITFMF 2013), in particular, to document each production task by different elements, namely (i) input data, (ii) input parameter, (iii) throughput, (iv) output, and (v) process metric. These recommendations are followed closely in the Generic Statistical Data Editing Models (UNECE 2015). In the present work, we will leave out the fifth element about the metric. We propose the following structure for every data-processing production task. We conceive every data-processing production task as a transforming action on a data set under a set of parameters producing a new data set or a new parameter set. We represent this as

$$\text{OutputData},\ \text{OutputParameters} := \text{Action}(\text{InputData},\ \text{InputParameters})$$

It must be noted that the distinction between data and parameter is somewhat arbitrary, since it depends on the semantic context of the concrete computation. For example, in Predict(InputData, PredictParameters) we compute predicted values for those data in the object InputData according to those parameters specified in the object PredictParameters, for instance, an ARIMA time series model ARIMA(p, d, q). Previously, we would

need to compute the degrees *p, d* and *q*. These can be computed similarly by PredictParameters: = ComputeDegrees(PredictParameters, DegreeParameters), where an initialised parameter object PredictParameters is updated with the computed degrees and where DegreeParameters specifies the parameters needed to compute *p, d* and *q*. Notice how in this second computation PredictParameters acts as an input data object.

This distinction concerning data and parameters can also be discussed in other common settings in standard production conditions. For instance, when joining two data sets, we can consider both data sets as elements of a more complex InputData object and the join resulting from the parameters specified in the corresponding InputParameters object (inner, outer and so on.). In the same vein, adding new records to an existing data set can also be modelled through a complex InputData object with an appropriate InputParameters object. Depending on the traceability and auditability provided to the whole system, the transient transformations can be further conveniently stored specifying timestamps, usernames and so on.

All in all, functional modularity principles can be used to implement this combination of paradigms by setting up a hierarchy of layers from (i) the statistical methodology, over its implementation in (ii) low-level procedures (possibly assembled in libraries) and (iii) high-level procedures thereof, to (iv) a process-orchestrating layer working as a user interface.

Notice how this organisation in layers also coincides with different traditional profiles at statistical offices. Statistical methodology is under mathematicians' and methodologists' responsibility, possibly also with the collaboration of domain experts. This layer focuses on the more abstract and mathematical part of the production system. The second layer implements the methodology as low-level software procedures. It falls under developers' and programmers' responsibility, possibly with the collaboration of programming-skilled methodologists. This layer still maintains a certain degree of abstraction. Concrete applications and production activities are shaped in the third layer under the responsibility of statisticians and survey managers, possibly with the aid of developers. In this layer, the collection of standard low-level procedures is adapted to the concrete needs of each statistical program. Finally, a process orchestrator working as user interface for ease of the human-computer interaction can additionally be put into place. This ease of use allows the management to optimise the production resources by potentially assigning tasks to non-specialists who follow previously specified protocols.

In the following sections we will use concrete surveys conducted at the Spanish National Statistics Institute to illustrate how this information architecture has been partially deployed for the statistical data editing phase. Our first step has been to propose a common data structure for all survey and administrative data sets (thus either InputData or OutputData) based on a standardised abstract data model for any kind of statistics. This is detailed in Section 4.

Next, we have implemented the optimisation-based selective editing techniques formerly developed at the Spanish National Statistics Institute (Arbués et al. 2013) following these principles. This boils down to designing and programming Actions together with different sets of InputParameters (also OutputParameters). We undertake this in Section 5.

4. Data Organisation

We will use the Spanish Retail Trade Survey and Service Sector Indicators Survey, conducted monthly at the Spanish National Statistics Institute, to illustrate the application

of this approach. These are short-term business statistics. Data are collected through paper questionnaires, telephone, fax, email, and CAWI modes. Statistical units are selected according to a stratified simple random sampling design. Target aggregates are mainly Laspeyres indices of both turnover and number of employees, possibly broken down into economic sector code and type of employment contracts, respectively.

In the preceding framework, our first task is to define an abstract data model for all statistical operations. The immediate goals of this model have been the versatility among all kinds of survey or administrative data and fast and easy deployment in the implementation.

The model essentially consists of a key-value pair data model, in which the key is composed by using the structural statistical metadata of the production system. We must distinguish between the data model for storing data in a corporative internal repository (the key is not parsed) and the data model for processing (the key is parsed). For manageability and rapid deployment reasons, in the current implementation the information is stored in plain text files, as explained below. These files are not modified once written. Updated information, if any, is included as a new file (with updated key in the name of the new file; see below). Concurrency issues and many other data architecture details are not considered relevant at this point.

The central element in the data model is the composition of the key for each single datum in the global production system at the office scale (or the whole statistical system scale). The key is composed of the following components:

(i) An alphanumerical code to identify the survey/statistical program.
This alphanumerical code is taken directly from the Spanish National Statistical Plan, where each survey/statistical program is univocally identified. This code references the concrete statistics where this value is generated, processed, and used.

(ii) An alphanumerical code to identify the time period of reference (coincidental with the time period of the corresponding statistics).
An ad-hoc simplified syntax has been put into place to denote the different reference time periods for all statistical operations according to the following table:

Time period	Code
Month	MM, MR
Trimester	TT, TR
Semester	SS, SR
Year	AA, AR

The second character denotes whether it is an ordinary data set or a duplicated data set containing statistical units from the rotated sample. This is especially used in short-term business statistics that use chain-linked Laspeyres indices with rotating panels.

(iii) An identifier to indicate whether they are raw or (partially) edited microdata, paradata, identification data and so on.

The different codes are:

Data file type	Code
Finally validated values	FF
Partially edited values	FD
Raw values	FG
Paradata	FP
Identification variable values	FI
Edit rules (Longitudinal phase)	FL
Edit rules (Cross-sectional phase)	FT

(iv) A version number either with the prefix *P* for provisional or *D* for definitive values.

(v) An identifier for the statistical variable.
This identifier is taken from the system of structural metadata so that each concept measured with a statistical operation in the whole statistical production system is identified with a standard name. For example, the concept of "turnover" is measured in different surveys (industry, retail trade, service sector and so on) and the same identifier *Turnover* is used in every survey. Subtleties in this statistical variable arising from its concrete usage in a survey is further specified using qualifiers (see immediately below).

(vi) A set of qualifiers specifying different attributes (statistical unit ID, geographical code, economic activity code and so on).
Qualifiers are variables that further specify the semantic content of each value. Although from a strict computer point of view, all qualifiers play the same role, this is not the case from a statistical standpoint. There are basically two types of qualifiers, namely, those that allow us to identify the statistical units, and the rest of them. The latter can be further divided into two categories. Firstly, as in the example below, there are qualifiers that amount to codes of standard classifications, such as the NACE, PRODCOM, COICOP and so on. At the Spanish National Statistics Institute, to the extent that it is feasible, international standard classifications are in use, in agreement with the ESS. In parallel, not all qualifiers of this type can be found in standard classifications. In these cases, in agreement with domain experts, the metadata unit puts into place a collection of internal standard classifications for these qualifiers. For example, the number of employees in a business unit is an extensively requested variable, usually broken down according to diverse criteria: by type of contract, by professional situation, and by type of remuneration. These have given rise to classifications with their own codes, which are used as qualifiers in the corresponding key. Secondly, there are qualifiers that are not necessarily understood as part of a classification. For example, the economic activity code of a business unit may change because of a change in its business activity, so that this variable in the population frame should be modified after receiving the updated information during field work. A qualifier (say, IsMod) denoting whether we are referring to the former value (IsMod $= 0$) or the modified value (IsMod $= 1$) must be introduced. This self-evident qualifier

value is not part of a classification. More specific qualifiers can always be used according to the specific process being executed. For example, in statistical data editing qualifiers in terms of population, measurement time, measured unit, and measured element can be properly defined, coded, and used as qualifiers (van der Loo 2015).

The following simplified example clarifies the meaning of these components. Let us consider the validated value of the turnover for a business unit (statistical unit ID 289409300MM) in the Retail Trade Survey (code E30103) in the reference time period of January 2016 in the region of Castilla-La Mancha (geographical code 08), in the economic sector of trade of food and beverages (NACE Rev.2 code 47.11). This value pertains to the first definitive data set for this time period. This is visually depicted in Figure 2. Note that some qualifiers are missing in this simplified example, as structural metadata defining the variable type (integer value expressed in euros).

As stated above, in the current implementation, data are stored in files. Each one is identified by statistical operation code, type of data (finally validated data, raw data, paradata, and so on), reference time period, and definitive or provisional character of the data in the production process. In other words, the common part of the key for a data set is encoded in the name of the corresponding file, where the rest of the key and the values are stored. In each file, each line will keep the standardised identifier and the rest of qualifiers together for each value (e.g., Turnover@@289409300MM47.1108@@9732 in our example). Other implementations are also possible.

A data dictionary is also configured and stored, containing the specifications of each statistical variable: name, description, data type – numeric or alphanumeric, maximal

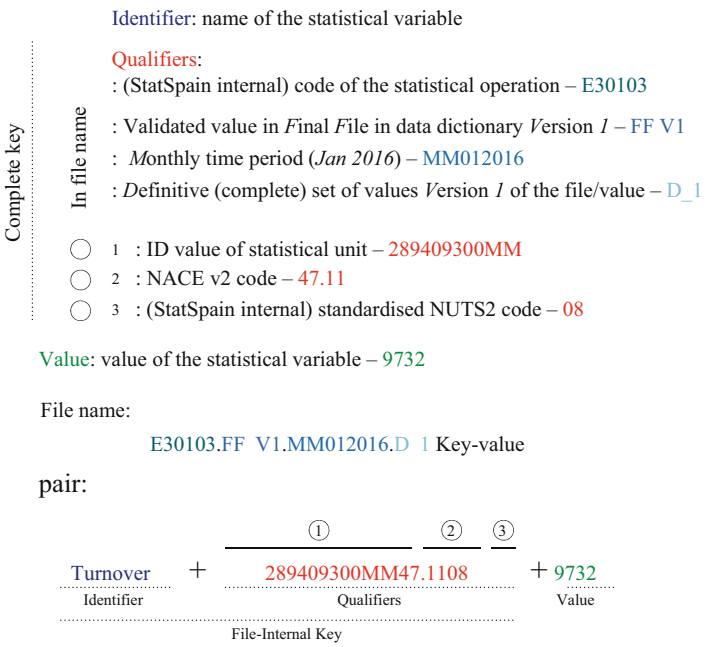

Fig. 2. Example of a key-value pair with a key composed of structural statistical metadata.

length – in terms of number of characters, qualifiers, corresponding domain-used variable names, range of values, and some other technical information for data collection applications. This dictionary allows the user to parse the key to instantiate objects according to a business logic class for all data processing tasks, which is indeed a data frame where the parsed key components are assigned in respective columns together with the corresponding value column. In this way, data are tidy, in the sense of Wickham (2014), for further processing with standardised transformations. Tidy data mean Codd's 3rd normal form so that (i) each variable forms a column, (ii) each observation forms a row, and (iii) each type of observational unit forms a table (see Wickham 2014). This business logic class consists essentially of the data frame and the data dictionary. Data transformations are applied on this class of objects, returning updated objects of the same class.

Immediate benefits are obtained after adopting such a data organisation. Firstly, since every data of every survey/statistical program can be managed in this way, a unique data architecture can be adopted throughout the entire production system of the office. This is a first crucial step towards the suppression of the stove-pipe production model, paving the way for a more efficient architecture. Having a common data architecture allows us to build standardised applications valid for all surveys, thus leading to the rationalisation of resources.

Secondly, these data specifications can be adapted to many actual circumstances in daily production. Let us consider, for instance, the case in which the economic activity code in the example changes along the process because the business unit has changed its activity. The example depicted here is oversimplified for ease of illustration. In practice, the metadata system has dozens of standard classifications for qualifiers (always international when possible) to parameterise each single datum along the process. In particular, we have four classifications that aim to pinpoint (i) the process stage in which the value is generated (design, collection, processing, dissemination, and so on, or a subprocess thereof), (ii) the element of the process which the value is related to (frame population, sample, questionnaire, and so on or a sub-element thereof), (iii) the role of the related actor in the process (statistical unit, interviewer, editing clerk, and so on), and (iv) the type of value (dichotomic variable, excluding variable, percentage, and so on). The evolution of the value along the process can be followed using these qualifiers. The metadata unit has put in place and is maintaining over 70 classifications and growing, as more statistical programs incorporate this architecture. Many classifications are very specific for a given statistical domain, but many others refer to features common to a large number of surveys.

Thirdly, the use of metadata in composing the keys to identify data values paves the way for achieving a standardised production system. In this way, every single datum in the whole production process is parameterised using, so to say, a common system of coordinates. In contrast to the dangerously common opinion of only conceiving metadata as a cumbersome documenting tool independent of production tasks and effective only after production has been executed (socalled *passive metadata* according to Lundell 2013), this data organisation makes use of the metadata system from the very beginning, in which data are generated and provide an interface between data and the user (*active metadata* according to the same author). Notice how this active role of metadata is key in

the sequence of transient transformations along the production process. Every independent transformation on a given data set must be implemented depending only on the input data and input parameters, that is, on the data and metadata contents that transform according to the parameters. If metadata are erroneous, the interface between data and the user is lost, and the process (as a sequence of transformations) cannot be executed.

5. Process Design

The design of the process architecture according to the principles set out in Section 3 is much more complex than the design of the data architecture. To begin with, a standard class of parameters (InputParameter) for all possible statistical methods (Action) is virtually impossible, since there exists a vast number of different statistical techniques. Thus, we will illustrate the application of the functional modularity principles with the concrete example of the optimisation approach to selective editing developed at the Spanish National Statistics Institute (see Arbués et al. 2013).

The division in layers begins by considering the statistical methodology at the bottom of the hierarchy. We will not go deep into the mathematical details and shall focus on the implementation of a very concrete formula to assign local (item) scores to each statistical unit.

The core of selective editing techniques is based on the assignment of a score to each variable to be edited for each statistical unit, thus providing a measure of the degree of suspicion of it containing an influential measurement error. The heuristic approach (de Waal et al. 2011) recommends choosing local (item) score functions such as $s_k = \omega_k |y_k - \hat{y}_k|$, where ω_k stands for the sampling weight of unit k and y_k, $\hat{y}_k$ denote the reported and predicted (expected) values of the variable y under editing, respectively. The main methodological content of the optimisation approach firstly consists of modelling the measurement errors $\epsilon_k = y_k - y_k^{(0)}$ ($y^{(0)}$ denoting the true value) for each unit and computing their first- and second-order moments M_{kl} for each pair of statistical units k and l (business units in our example) and each variable y (turnover and number of employees in our example). These are given by analytical expressions (Arbués et al. 2013):

$$\begin{aligned} M_{kk} &= \sqrt{\frac{2}{\pi}}\,\omega_k\,\hat{\nu}_k\,{}_1F_1\left(-\frac{1}{2};\frac{1}{2};-\frac{(y_k-\hat{y}_k)^2}{2\hat{\nu}_k^2}\right)\zeta_k\left(\frac{y_k-\hat{y}_k}{\hat{\nu}_k}\right), \\ M_{kl} &= 0, \quad k \neq l, \end{aligned} \tag{1}$$

for the loss function $L(a,b) = |a-b|$ and

$$\begin{aligned} m_k &= \omega_k\,\hat{\nu}_k\,\frac{\hat{\sigma}_k^2}{\hat{\sigma}_k^2+\hat{\nu}_k^2}\left(\frac{y_k-\hat{y}_k}{\hat{\nu}_k}\right)\zeta_k\left(\frac{y_k-\hat{y}_k}{\hat{\nu}_k}\right), \\ M_{kk} &= \omega_k^2\,\hat{\nu}_k^2\left(\frac{\hat{\sigma}_k^2}{\hat{\sigma}_k^2+\hat{\nu}_k^2}\right)^2\left[\frac{\hat{\sigma}_k^2+\hat{\nu}_k^2}{\hat{\sigma}_k^2}+\left(\frac{y_k-\hat{y}_k}{\hat{\nu}_k}\right)^2\right]\zeta_k\left(\frac{y_k-\hat{y}_k}{\hat{\nu}_k}\right) \\ M_{kl} &= m_k\,m_l, \quad k \neq l, \end{aligned} \tag{2}$$

for the loss function $L(a,b) = (a - b)^2$, where in both cases

$$\zeta_k(x) = \frac{1}{1 + \frac{1-\hat{p}_k}{\hat{p}_k}\left(\frac{\hat{\nu}_k^2}{\hat{\sigma}_k^2 + \hat{\nu}_k^2}\right)^{-1/2} \exp\left(-\frac{1}{2}\frac{\hat{\sigma}_k^2}{\hat{\sigma}_k^2 + \hat{\nu}_k^2}x^2\right)}$$

Exact details about the derivation of expressions (1) and (2) are given by Arbués et al. (2013). In the first case, when $|\frac{y_k - \hat{y}_k}{\hat{\nu}_k}| \to \infty$, $M_{kk} \to \omega_k |y_k - \hat{y}_k|$, which is the usual expression in the heuristic approach (de Waal et al. 2011) (in this case M_{kk} can be viewed as item scores). Thus, Formulas (1) and (2) can be understood as a rigorous generalisation of the traditional approach to selective editing by using statistical models for measurement errors. The scores also depend on other parameters, such as the probability of reporting an erroneous value and the variability of these errors reported in the past. As a matter of fact, statistical models for the measurement error are behind the diverse parameters in these expressions:

- ω_k denotes the sampling (design) weight of unit k;
- y_k denotes the raw (reported) value of variable y for unit k as collected in the questionnaire;
- $\hat{y}_k$ and $\hat{\nu}_k$ denote the predicted value and its prediction standard deviation for variable y and unit k;
- $F_1(x;y;z)$ stands for the confluent hypergeometric function of the first kind (Pearson et al. 2017), which arises from the choice of the loss function in the underlying optimisation problem;
- $\hat{p}_k$ denotes the estimated probability of measurement error for variable y and unit k, that is, $p_k = \mathbb{P}\left(y_k \neq y_k^{(0)}\right)$, where $y_k^{(0)}$ stands for the true value of variable y;
- $\hat{\sigma}_k^{(0)}$ denotes the estimated standard deviation for the observed measurement error $Q_k = y_k - y_k$.

These quantities can be computed for the whole population or by population cells (e.g., determined by economic sector or geographical region, or both).

Now we consider the second and third layers, in which the statistical methodology is implemented in finer – and coarser – grained production tasks. From the methodology, it is clear that the error moments can be written as functions of diverse parameters $M_{kl} = M_{kl}(y_k, \hat{y}_k, \hat{\nu}_k, \hat{\sigma}_k, \hat{p}_k, \omega_k)$. Now the question arises regarding how to organise this computation in a modular way.

At this point, functional modularity and statistical methodology must be precisely combined. From a strictly computational point of view, there is no distinction between the parameters y_k, $\hat{y}_k$, $\hat{\nu}_k$, $\hat{\sigma}_k$, $\hat{p}_k$, and ω_k. However, from a statistical point of view, this distinction is fundamental for allowing the system to efficiently grow and evolve in the future. Raw values y_k are taken directly from the data collection stage. Independent modules will handle the computation of $\hat{y}_k$ and $\hat{\nu}_k$ (prediction module), $\hat{\sigma}_k$ (observation error estimation module), $\hat{p}_k$ (error probability estimation module), and ω_k (sampling design module). The computation of these parameters will be completely independent of each another and each one will depend exclusively on its input arguments. They will

interact with each other only through their final computed values, so that the computation is transparent.

This organisation in modules is justified by the underlying statistical knowledge. First, there are many prediction methods that potentially could be applied to obtain both $\hat{y}_k$ and $\hat{v}_k$. If new methods need to be added to the system, this can be done without affecting the rest of the computation. This same observation is valid for the remaining modules. Note that this is a simple example in which we are computing a single value with an analytical formula with just six arguments. The consequences of a poor (from a methodological point of view) modular organisation, may produce the opposite effect across the entire production system. This is why functional modularity and statistical methodology must be precisely combined in the design of the production system.

Each module, in turn, makes use of these same principles, so that different methodological aspects of the computation are considered independently. For example, due to missing values or some other reason, predicted values cannot be computed for all statistical units and must be imputed. An independent module for imputation is thus constructed to handle this task independently of any other, and embedding it in the former computation. The architecture is, again, the same:

$$\text{ImputedObject} := \text{Impute}(\text{InputObject}, \text{ImputationParameters}).$$

The whole computation is then constructed as follows. Firstly, the Action element specifying the concrete production task will be denoted by ComputeErrorMoment and will implement either Formula (1) or (2), depending on its arguments.

As InputData we set all elements in Expressions (1) and (2), namely (i) the values of the target variables y (turnover and number of employees in our example), (ii) some other auxiliary variables (e.g., those determining different population domains; economic classification NACE codes, and Spanish geographical classification codes in our example), and (iii) the model parameters $\theta_k = (\hat{y}_k, \hat{v}_k, \hat{\sigma}_k, \hat{p}_k, \omega_k)$ for each variable y and each unit k. These are the parameters for the continuous variable observation-prediction model (Arbués et al. 2013). We will call this InputData data set contObsPredModParam and it is given the key-value pair structure described in the preceding section. These parameters (hence the object contObsPredModParam) must be computed with their respective modules:

- The predicted values $\hat{y}_k$ and their standard deviations $\hat{v}_k$ are computed by initialising the object contObsPredModParam and defining an abstract class PredictionParam for the input parameter. The computation is carried out by updating the object contObsPredModParam:

  $$\text{contObsPredModParam} := \text{ComputePred}(\text{contObsPredModParam}, \text{PredictionParam}).$$

  The concrete statistical method used to compute $\hat{y}_k, \hat{v}_k$ is specified by defining a concrete subclass of PredictionParam. In our example, we have defined four time series models (random walks with regular, seasonal, and regular/seasonal differences and automatic ARIMA models), among which the one with the lowest $\hat{v}_k$ is automatically selected. Any alternative choice (e.g., with machine learning techniques) could easily be implemented by defining the corresponding subclass.

Hierarchy and layering principles are applied by internally constructing routines on the key-value pair data structure in terms of simpler data structures such as vectors. In addition, imputation routines can be embedded in the design of these classes and methods as an attribute of PredictionParam.

- The estimated standard deviation $\hat{\sigma}_k$ of observation errors is computed in the same way:

$$\text{contObsPredModParam} := \text{ComputeObsErrorSTD}(\text{contObsPredModParam}, \text{ObsErrorSTDParam}).$$

In this case, another abstract class ObsErrorSTDParam has been defined. Its concrete subclasses determine the statistical method to be used for the estimation. In our example, we have defined a subclass to estimate σ_k by maximum likelihood, using the historical double sets of raw and validated data. As before, imputation routines can also be embedded in the design of these classes and methods as an attribute of ObsErrorSTDParam.

- The estimated error probabilities $\hat{p}_k$ are also computed in the same way:

$$\text{contObsPredModParam} := \text{ComputeErrorProb}(\text{contObsPredModParam}, \text{ErrorProbParam}).$$

In this case, an abstract class ErrorProbParam is defined. Its concrete subclasses determine the statistical method to be used for the estimation. In our example, we have defined a subclass to estimate p_k by maximum likelihood, using the historical double sets of raw and edited data. Again, as before, imputation routines can also be embedded in the design of these classes and methods as an attribute of ErrorProbParam.

- The sampling weights ω_k are usually computed at an earlier stage of the production process, so that we can simply retrieve them from some other data set in the survey in question. In other cases, if explicitly needed for the editing phase, the computation of the sampling weights can be carried out along similar lines.

Next, as parameters InputParameter in our error moments computation, we essentially need to specify the loss function $L(\cdot,\cdot)$. We will denote this object by ErrorMomentParam.

Finally, the output object OutputData will be denoted by ErrorMoments and is basically an array of error moment matrices $[M_{kl}^{(q)}]$ per population cell (q denotes the turnover and the number of employees in our example). In this way, we already have the content of each object in the expression

$$\text{ErrorMoments} := \text{ComputeErrorMoment}(\text{contObsPredModParam}, \text{ErrorMomentParam})$$

The whole computation at the third (scripting) layer is thus executed just by calling something like

$$\begin{aligned} \text{DD} &:= \text{readFile}(\text{DataDictionaryFile}) \\ \text{contObsPredModParam} &:= \text{buildcontObsPredModParam}(\text{DD}) \\ \text{PredictionParam} &:= \text{buildPredictionParam}(\text{and so on}) \end{aligned}$$

```
contObsPredModParam := ComputePred(contObsPredModParam, PredictionParam)
ObsErrorSTDParam := buildObsErrorSTDParam(and so on)
contObsPredModParam := ComputePred(contObsPredModParam, ObsErrorSTDParam)
ErrorProbParam := buildErrorProbParam(and so on)
contObsPredModParam := ComputePred(contObsPredModParam, ErrorProbParam)
SamplingWParam := buildSamplingWParam(and so on)
contObsPredModParam := ComputePred(contObsPredModParam, SamplingWParam)
ErrorMoments := ComputeErrorMoment(contObsPredModParam, ErrorMomentParam)
```

In the construction of the diverse parameters objects, the same hierarchical scheme can be followed (including e.g., the imputation routines). Notice also the far-reaching consequences on the organisation of work and the production process at a statistical office. Firstly, survey managers and domain experts can work at a scripting level with high-level functions such as ComputePred, ComputeObsErrorSTD, and ComputeErrorProb above. This does not demand extensive IT skills and they can concentrate on the adapted use of these tools to their concrete survey. Indeed, the modularity allows them to seamlessly combine and choose diverse alternatives to compute the parameters and the error moments according to the characteristics of the statistical operation. On the other hand, developers and methodologists (ideally data scientists) can work at a lower level, implementing new statistical methods as new subclasses and overloaded methods. Needless to say, for an optimal design of classes and methods, communication between both layers is recommended. Notice however that both the naming conventions and the division in modules (both functions and libraries) derives directly from the application of the foregoing principles: it is the statistical methodology which should define the borders (interfaces) between the different modules. This paves the way for easy application of standard good practices in software development, supported by a strong mathematical basis. In the current development and implementation of our proposal, we can only offer an empirical view on this particular production stage (editing). However, if these principles are to be applied throughout the process, the different functional modules should similarly interface with one another, thus coping with complexity.

Secondly, this architecture favours software evolution and ease of maintenance over code preservation (Booch et al. 2007). Legacy code is recognised as a heavy ballast in the modernisation of statistical production. We are not providing solutions for the existing legacy code, but this architecture philosophy helps a great deal by not producing legacy code. The code can evolve according to new needs detected in the statistical programs, by defining new subclasses and methods. At the same time, the produced code is easily maintained, since execution statements such as the one above seldom change.

Thirdly, since statistical methods are implemented with an abstraction of concrete statistical operations, the same code at the lower level and very similar at the scripting level is valid for different surveys. This allows us to optimally manage resources among statistical operations, as the methodology and computer tools are standardised.

Fourthly, we would like to comment on the granularity of the services and computer tools. In our example above, by starting with Formulas (1) and (2), we also want to suggest that the statistical methodology should determine the degree of granularity of computer

tools that implement the different methods. In the modular design, the statistical methods themselves should determine the natural borders among modules (hence also their interconnecting interfaces). Furthermore, the internal components of each module should also be structured according to the statistical methodology. Note how, in our example above, each parameter entered into Formulas (1) and (2) is dealt with using an independent method on the object contObsPredModParam, because each parameter can be computed/estimated choosing an adequate statistical method. Should new methodological proposals appear for a concrete computation, these can easily be incorporated without affecting the other software routines (e.g., imputation routines).

Finally, we would like to underline how the scripting philosophy fits perfectly well in the GSDEMs as a processing step, in which input statistical data and input metadata, process details, and transformed statistical data and output metadata are clearly expressed (UNECE 2015). Although we have not yet used this process architecture to manage process metrics, we are convinced that these monitoring parameters can also be computed along similar lines. This may be carried out by complementing each computation or transformation on an input data set with a chosen set of indicators in the output monitoring the transformation.

In conclusion, we must mention that in addition to the foregoing technical, mathematical difficulties, a highly relevant element of the practical implementation of this proposal is the staff reaction to changes in the production system. In the current stage of prototyping in production in a few statistical operations, the role of survey managers has been identified as key, since, in our current production model, they take the decisions on each survey. The gap between statisticians and computer scientists (and their traditional skills) also stands out as an aspect that needs to be addressed further.

6. Implementation: A Proof of Concept

The principles of functional modularity have been applied by designing and developing independent software packages for concrete aspects of this data organisation and process design. There are many aspects of the implementation worth sharing in order to be acquainted with the interplay between theoretical proposals and the practicalities arising in an ongoing production system at a statistical office.

Firstly, since both object-oriented and functional paradigms lie at the core of the proposal, the natural choice for a programming language is one that naturally supports these paradigms, without syntax quirks and twists. Java, C++, R, Python, Scala and many others are candidates that fulfill this condition. Since the user domain is clearly statistical data processing, another requisite is feasible rapid development of trustworthy statistical tools. Finally, a good documenting system of classes, methods, and functions is also desirable, which allows us to document data and parameter inputs, output, and throughput of each element (the process statistical metadata). These considerations led us to choose R (R Core Team 2012; Chambers 2008).

Secondly, the methodology of software development has also been carefully decided. Instead of the more classical waterfall model (see e.g., Palmquist et al. 2013), we have used a spiral approach (Boehm 1988). Thus, instead of compiling specifications, designing, coding, and testing in a linear way, we have incrementally agreed on a first

round of specifications, made a first design implemented on a first version of several R packages, and constructed a first version of the repository with key-value data files for three different short-term business statistics surveys. In this first round, the physical layer (the files themselves), the programming layer (classes, methods, and functions: the R packages), and the scripting layer were constructed. In a second round, apart from bugs and flaws in some functions detected in the testing phase, an important redesign was discovered to be necessary in the classes and methods implementation. The technical reason was that, for performance reasons in order to handle these key-value pair data sets, our packages heavily depend on the package data.table (Dowle and Srinivasan 2016). Formerly we used the S4 system of classes and methods, and the method dispatch, which suspends the lazy evaluation, is thus incompatible with the data.table syntax. We migrated all key-value data packages to the system S3. This affected the second layer, and interestingly enough, it did not affect the scripting layer. Along this line of work, we pursue the production of constantly evolving pieces of software that can adapt quickly and straightforwardly to the needs and changes of production. Again, this change of philosophy is at odds with the traditional culture at a statistical office and requires formidable management efforts in order to implement it at the officewide scale. For example, the idea that computer tools built in this way are not completed and ready for use in production may be risky, since it may lead to rejection of the methodology due to immature tools. These, more agile, methodologies also allow us to make more rational use of scarce resources, since development is incremental. In our view, a mindset change to perceive software as constantly evolving, rather than as a closed definitive tool is necessary for the industrialisation and modernisation of statistical production.

Thirdly, as a byproduct of the preceding methodology, communication between domain experts and survey managers, on the one hand, and developers and methodologists, on the other hand, must be clearly stressed. Although the architecture makes the work of both profiles independent by defining programming and scripting layers, an optimal system design will be achieved when communication between both parts is at a maximum during the development stage. Again, we face a management challenge that may impinge on organisational aspects of the whole statistical office (e.g., does it make sense to differentiate between statistical methodology and statistical software development departments?).

Fourthly, the different actors' computer skills must be taken into account. Two further actions have been taken in this regard to deploy the preceding architecture at the Spanish National Statistics Institute. On the one hand, the file containing the data dictionary is an XML file for machine readability. This technology does not form part of regular computer skills of domain experts and survey managers. Thus, to build this file, we asked these statisticians to record the specifications of each statistical variable in their survey in an Excel file with a prespecified structure. Excel files, although limited when dealing with some data structures, are easily handled. Then, we programmed a specific function, building the data dictionary file automatically from this Excel file.

Fifthly, the statistical computing system used as a standard at the Spanish National Statistics Institute is SAS, and following this institutional policy, computing routines used by survey managers and domain experts must be written in SAS, and not in other languages such as R, Python, Scala, and so on. Thus, the fourth layer, working as a user

interface, has been developed in the form of extremely simplified SAS macros that execute the aforementioned R scripts in batch form. This means that the interaction between the user and the architecture occurs only in SAS (so far, this has only been accomplished to feed and read from the repository; the selective editing routines are executed directly by data collection staff in simplified R scripts). Although the functionality of the system is currently severely reduced and rigidity is increasing, ease of use is noticeable, as the user only needs to specify a few very generic parameters.

Finally, the collection of packages in constant evolution at various stages of maturity are available in GitHub (Esteban et al. 2017a,b,c,d,e,f,g,h,i,j,k,l; Sanguiao 2017). The architecture behind these packages closely follows the statistical methodology of the optimisation approach to selective editing. Thus, it is difficult to give a precise description of what each package does without entering into mathematical content. A summarised description of what each package does can be found in Esteban et al. (2017m). It is important to point out that this division into many different packages that focus on concrete aspects of the statistical process should not be read just as an example of good practices in programming, but also as a consequence of the identification of functional modules according to the underlying statistical methodology.

7. Conclusion and Future Prospects

The main conclusion from this work is that in recognising an official statistical production system as a human-generated complex system, the principles of functional modularity can be used to cope with this complexity of designing both data and process architectures adapted and adaptable to the evolving needs of statistical production. By moving a step in the direction of international standards, we can combine the object-oriented and functional paradigms to define functional modules for the different production tasks whose borders and interacting interfaces are naturally determined by means of the underlying statistical methodology. These principles drive us genuinely to a set of layers in the statistical methodology, over its implementation in lower – and higher – level production tasks and steps to a top-orchestrating user interface.

The data organisation essentially revolves around a key-value pair data model, where keys are composed of statistical metadata. The process architecture implements transformations over information objects, thus combining both paradigms. In our view, these architectures bring relevant benefits to an efficient production system. They provide due roles for the different professional profiles in a statistical office, favour the evolution of software, thus adapting to new needs, lead to complete global parametrisation of every single datum in the process, and lead to standardisation in the production tools in surveys and statistical programs of various types.

Some of the elements presented in preceding sections are connected with the concrete production system at the Spanish National Statistics Institute. Therefore, it is advisable to recognise those elements that are exportable to other offices. Regarding the data architecture, the core element is the use of metadata to identify values. The key-value pair structure could be substituted by alternative data models, such as the SDMX or DDI. Nonetheless, in a deeper stage of analysis, performance issues (among others) should be taken into account in making a choice. In our case, we can process monthly data sets of

around 2 million lines and about 15 qualifiers (around 28,000 business units) to construct their corresponding traditional data matrices in less than two seconds. Regarding the process architecture, the core elements are (i) the application of functional modularity to statistical methods to produce modular computations respecting the natural borders in statistics, (ii) the layers organising the production tasks at different degrees of modularity, (iii) the use of object-oriented modelling for the information objects (both data and parameters), and (iv) the use of the functional paradigm to carry out the chained transformations on these information objects. All other implementation details can be adapted to concrete circumstances.

Nonetheless, our proof of concept reveals relevant challenges ahead. To be more efficient, an agile software development methodology should be preferred over more static methodologies. Also, it is important that the existing gap between methodologists/statisticians and computer scientists/developers must be bridged. All this pushes us to improve communication standards among the different actors (methodologists, computer scientists, domain experts, survey managers, business managers, and so on) within an office. This a remarkable management exercise.

Along this line, as stakeholders in and members of the ESS we recognise that alignment with international initiatives is a strategic matter. Thus, in future revisions and developments, alignment with CSPA services and European standards will be taken into account and pursued. Previously, technical requisites to be CSPA-compliant and to achieve shareability of computer application must be agreed upon by the international community (see, for example, the 2017 meeting report of UNECE 2017a).

8. References

Arbués, I., P. Revilla, and D. Salgado. 2013. "An optimization approach to selective editing." *Journal of Official Statistics* 29: 489–510. Doi: http://dx.doi.org/10.2478/jos-2013-0037.

Boehm, B. 1988. "A spiral model of software development and enhancement." *IEEE Computer* 21(5): 61–72. Doi: http://dx.doi.org/10.1145/12944.12948.

Booch, G., R.A. Maksimchuk, M.W. Eagle, B.J. Young, J. Conallen, and K.A. Houston. 2007. *Object-oriented Analysis and Design with Applications*. Addison-Wesley.

Chambers, J.M. 2008. *Software for Data Analysis*. Springer.

DDI Alliance. 2018. *Data Documentation Initiative 2018*. Available at https://www.ddialliance.org/ (accessed November 05, 2018).

De Waal, T., J. Pannekoek, and S. Scholtus. 2011. *Handbook of Statistical Data Editing and Imputation*. Wiley.

Dowle, M. and A. Srinivasan. 2016. *data.table: Extension of 'data.frame'*. Available at https:// CRAN.R-project.org/package=data.table. R package version 1.10.0.

Esteban, E., S. Saldaña, and D. Salgado. 2017a. *RepoTime: Implementation of a notation for time intervals*. Available at https://github.com/david-salgado/RepoTime. R package version 0.2.2.

Esteban, E., S. Saldaña, and D. Salgado. 2017b. *StQ: Tools to manage metadata-incorporated keyvalue pair datasets*. Available at https://github.com/david-salgado/StQ. R package version 0.4.34.

Esteban, E., S. Saldaña, and D. Salgado. 2017c. *RepoReadWrite: Read and write files from/to the microdata repository*. Available at https://github.com/david-salgado/RepoReadWrite. R package version 0.4.5.

Esteban, E., S. Saldaña, and D. Salgado. 2017d. *RepoUtils: Implementation of tools to map and work with repositories*. Available at https://github.com/david-salgado/RepoUtils. R package version 0.1.2.

Esteban, E., S. Saldaña, and D. Salgado. 2017e. *contObsPredModelParam: Class and methods for the parameters of a continuous observation- prediction model*. Available at https://github.com/david-salgado/contObsPredModelParam. R package version 0.0.1.

Esteban, E., S. Saldaña, and D. Salgado. 2017f. *StQPrediction: Define S4 classes and methods to make predictions*. Available at https://github.com/david-salgado/StQPrediction. R package version 0.0.1.

Esteban, E., S. Saldaña, and D. Salgado. 2017g. *StQImputation: Classes and methods to implement different imputation methods upon StQ objects*. Available at https://github.com/david-salgado/StQImputation. R package version 0.0.1.

Esteban, E., S. Saldaña, and D. Salgado. 2017h. *SelEditErrorMoment: Compute the conditional measurement error moments under the optimization approach to selective editing*. Available at https://github.com/david-salgado/SelEditErrorMoment. R package version 0.0.1.

Esteban, E., S. Saldaña, and D. Salgado. 2017i. *SelEditFunctions: Functions for selective editing*. Available at https://github.com/david-salgado/SelEditFunctions. R package version 0.0.1.

Esteban, E., S. Saldaña, and D. Salgado. 2017j. *SelEditUnitPriorit: Classes and methods to implement unit prioritization*. Available at https://github.com/david-salgado/SelEditUnitPriorit. R package version 0.0.1.

Esteban, E., S. Saldaña, and D. Salgado. 2017k. *TSPred: Point and std prediction of time series*. Available at https://github.com/elisa-esteban/TSPred. R package version 0.2.5.

Esteban, E., S. Saldaña, and D. Salgado. 2017l. *BestTSPred: Construction of objects of class BestTSPredParam*. Available at https://github.com/elisa-esteban/BestTSPred. R package version 0.0.1.

Esteban, E., S. Saldaña, and D. Salgado. 2017m. Software implementation of optimization-based selective editing techniques at Statistics Spain (INE). UNECE Work Session on Statistical Data Editing. The Hague, 24–26 April 2017. Available at https://www.unece.org/fileadmin/DAM/stats/documents/ece/ces/ge.44/2017/mtg2/Paper_19_StatSpain.pdf (accessed November 05, 2018).

Eurostat. 2014a. ESS Vision 2020. Available at http://ec.europa.eu/eurostat/web/ess/about-us/ess-vision-2020.

Eurostat. 2014b. Vision 2020 Implementation Portfolio. Available at http://ec.europa.eu/eurostat/web/ess/about-us/ess-vision-2020/implementation-portfolio.

HLG-MOS. 2011. "High-Level Group for the Modernisation of Official Statistics. Strategic vision of the High-Level Group for strategic developments in business architecture in Statistics." *Conference of European Statisticians Geneva. 59th Plenary Session*. 14–16 June, 2011. Working Paper 1. Available at https://www.unece.org/fileadmin/DAM/stats/documents/ece/ces/2011/1.e.pdf.

HLG-MOS. 2017. High-Level Group for the Modernisation of Official Statistics. UN-ECE Statistics Wikis. Available at http://www1.unece.org/stat/platform/display/hlgbas/High-Level+Group+for+the+Modernisation+of+Official+Statistics.

Informal Task Force on Metadata Flows. 2013. "Metadata flows in the GSBPM." *Work Session on Statistical Metadata*. Geneva, 6–8 May, 2013. Working Paper 22. Available at https://www.unece.org/fileadmin/DAM/stats/documents/ece/ces/ge.40/2013/WP22.pdf (accessed November 05, 2018).

Lundell, L.-G. 2013. Framework of metadata requirements and roles in the SDWH. ESSnet on microdata linking and data warehousing in production of business statistics. Deliverable 1.1. Available at https://ec.europa.eu/eurostat/cros/content/dwh-sga2-wp1-11-metadata-framework-statistical-data-warehousing-v112-final_en.

Palmquist, M.S., M.A Lapham, S. Miller, T. Chick, and I. Ozkaya. 2013. Parallel worlds: agile and waterfall differences and similarities. Technical Note CMU/SEI-2013-TN-021. Software Engineering Institute. Carnegie Mellon University. Available at http://repository.cmu.edu/cgi/viewcontent.cgi?article=1761&context=sei.

Pearson, J.W., S. Olver, and M.A. Porter. 2017. "Numerical methods for the computation of the confluent and Gauss hypergeometric functions." *Numerical Algorithms* 74: 821–866. Doi: http://dx.doi.org/10.1007/s11075-016-0173-0.

R Core Team. 2012. *R: A language and environment for statistical computing. R Foundation for Statistical Computing*. Available at http://www.R-project.org.

Saltzer, J.H. and M.F. Kaashoek. 2009. "Principles of computer system design: An Introduction. Morgan Kaufmann, 2009. ISBN: 978-0-12-374957-4.

Sanguiao, L. 2017. *Transformation of Standard Questionnaires*. Available at https://github.com/Luis-Sanguiao/StQT. R package version 0.1.0.9000.

UNECE. 2013a. Generic Statistical Business Process Model. Version 5.0. Available at http://www1.unece.org/stat/platform/display/metis/The+Generic+Statistical+Business+Process+Model.

UNECE. 2013b. Generic Statistical Information Model. Version 1.1. Available at https://statswiki.unece.org/display/gsim/Generic+Statistical+Information+Model.

UNECE. 2015. Generic Statistical Data Editing Models. Version 1.0. Available at https://statswiki.unece.org/display/kbase/GSDEMs.

UNECE. 2017a. Statistical Data Editing Work Sessions. Available at http://www1.unece.org/stat/platform/display/kbase/UNECE+Work+Sessions+on+Statistical+Data+Editing.

UNECE. 2017b. Capabilities and Communication Group. Available at http://www1.unece.org/stat/platform/display/MCOOFE/Capabilities+and+Communication+Group%3A+Home.

Van der Loo, M. 2015. A formal typology of data validation functions. UNECE Work Session on Statistical Data Editing. Budapest, 14–16 September 2015. https://www.unece.org/fileadmin/DAM/stats/documents/ece/ces/ge.44/2015/mtg1/WP_5_Netherlands_A_formal_typology_of_data_validation_functions.pdf (accessed November 05, 2018).

Van Roy, P. and S. Haridi. 2004. "Concepts, Techniques, and Models of Computer Programming." MIT Press.

Weinberg, G.M. 2011. "An introduction to General Systems Thinking." Weinberg and Weinberg. ISBN: 978-0-93-263349-1.

Wickham, H. 2014. "Tidy data." *Journal of Statistical Software* 29(10): 1–23. Doi: http://dx.doi.org/10.18637/jss.v059.i10.

Received June 2017
Revised March 2018
Accepted June 2018