Working Paper.

UNITED NATIONS
ECONOMIC COMMISSION FOR EUROPE

CONFERENCE OF EUROPEAN STATISTICIANS

Work Session on Statistical Data Editing
(Neuchâtel, Switzerland, 18-20 September 2018)

A standardized approach to editing: Statistics Finland's metadata-driven editing and imputation service

Prepared by Rok Platinovsek, Statistics Finland

Abstract

A modern approach to editing and imputation (E&I) should be metadata driven, produce a full audit trail, enable results to be reproduced, and automatically produce standardized reports. This paper outlines a metadata information model wherein the parameters of E&I methods are described at three levels. Firstly, the most general level describes the *generic E&I method* and the various input parameter types it can use. Secondly, a *particular E&I method* is described by specifying the name and type of each parameter the method uses. Finally, ascribing values to the particular method's parameters gives rise to the *method instance*. It is the method instances that appear as steps in the E&I process flow and drive software.

We proceed to describe the metadata-driven editing service developed at Statistics Finland. The editing service is essentially passed a dataset to be edited and a reference to the process flow wherein a sequence of method instances is delineated following BPMN 2.0. The editing service relies on an external process management service to determine at each point in the process the method instance to execute next. The editing service then retrieves from the metadata repositories the method instance's parameter values and executes it. Having reached the end of the process flow, the editing service returns the edited dataset and an audit trail and produces standardized reports. The methods currently supported by the editing service include Banff methods and a SAS-implementation of the if-then rule method. The editing rules for the if-then rule method are specified using the Validation and Transformation Language and are translated to SAS before execution.

I. Introduction

1. Statistics Finland is undergoing a comprehensive modernization of its social statistics production under the umbrella project called STIINA (Social Statistics Integrated Information System). The aim is to model statistics production processes using the Generic Statistical Business Process Model (GSBPM; UNECE 2013a) and develop generic metadata-driven services that perform tasks identified by the GSBPM. The Generic Statistical Information Model (GSIM; UNECE 2013b), in turn, is relied upon to model the services' inputs and outputs.

2. Statistics Finland has committed to conducting its future software development in accordance to the so-called service oriented architecture. Rather than developing large "monolithic" software solutions, the idea is to break them up into small parts, each part performing a specific function. The services communicate with one another using standard protocols like HTTP. The benefits of using service-oriented architecture include allowing the services to be re-used in different contexts, isolating faults, and reducing long-term commitment to technologies as each technology-specific solution is enclosed in a technology-neutral wrapper.

3. One of the metadata-driven services whose development has been undertaken under the STIINA project is the editing service implementing functionalities defined in GSBPM phases 5.3 (Review and validate) and 5.4 (Edit and impute). Its requirements, information objects, and possible solutions have been investigated in a preceding feasibility study project (see Platinovsek 2017). The editing service has the goals of being technology-neutral and fully metadata-driven. It should produce standardized reports facilitating the calculation of quality indicators like the imputation rate and provide transparency and traceability of E&I actions via a detailed audit trail.

4. Before giving an account of the editing service itself, we will briefly describe the Banff system for editing and imputation, as certain solutions we have adopted were influenced by Banff and the Banff Processor (see Statistics Canada 2012, 2014).

II. Banff

5. Banff is a collection of SAS procedures used for editing and imputation. One of the strengths of the Banff system is that it is modular, meaning that the SAS procedures can be used independently of each other. The procedures can be combined somewhat freely to produce the desired E&I process flow, e.g., attempt to impute values with a preferred method and where that fails impute the remaining values with a more robust method (Statistics Canada 2014). The logical order in which Banff procedures are applied in the context of survey processing is given in Figure 1.

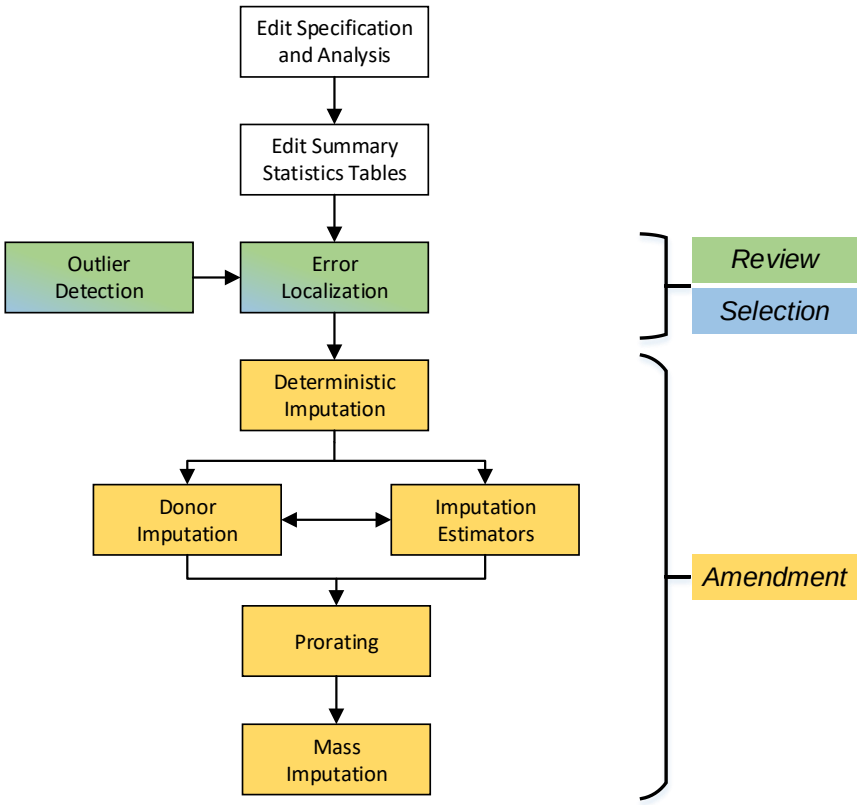

Figure 1: The logical order of the Banff procedures in the context of survey processing; source: Figure 1.1 (Statistics Canada 2014). Categorization with regard to purpose added by the author.

6. According to the Generalized Statistical Data Editing Models framework (GSDEMs, UNECE 2015, see also Pannekoek and Zhang 2012) E&I activities fall into three broad categories in terms of their purpose. Examining the data and trying to identify potential problems are classified as *review* activities. *Selection* refers to selecting units or variables within units for further treatment. *Amendment*, in turn, refers to actually changing selected data values in a way that is considered appropriate to improve the data quality.

7. Applying this classification to the typical process flow of Banff procedures in Figure 1 reveals an unsurprising pattern: Banff procedures implementing review and selection are applied first and are flowed by amendment procedures. Banff's outlier detection module can be seen as implementing both review and selection functions as it, firstly, assesses a data point's plausibility by calculating an outlier measure (review) and, secondly, marks up data points with measures exceeding a specified threshold for further treatment (selection). The error localization procedure, likewise, performs a dual function. It evaluates a record's validity according to a set of edit rules (review) and uses Fellegi and Holt's (1976) error localization approach to identify data points for treatment (selection of variables).

8. In the Banff system, selection is technically implemented by marking data points for further treatment in the so-called status dataset. This information is passed down to imputation procedures that actually change (amend) the data points' values. Each Banff amendment procedure requires such a status dataset as input, but also outputs a status dataset of its own wherein it flags the data points it amended.

9. The use of status datasets is a central feature of the Banff system. On the one hand, it allows review and selection procedures to pass information to amendment procedures downstream in the E&I process. This enables the Banff system to be modular. On the other hand, the status dataset partially contains the audit trail information as each amendment procedure stamps treated data points with its own status identifier. We note, however, that the audit trail contained in Banff's status datasets does not sufficiently document the E&I actions to allow their results to be reproduced. The status dataset merely identifies the applied methods but does not specify the methods' parameters.

III. Modelling the parameters of E&I methods

10. The following sections describe how modelling of E&I methods' parameters can benefit from introducing a three-part structure and how this scheme has been used in the concrete implementation of Statistics Finland's editing service.

A. Three layers of metadata

11. We have found it is useful to model the parameters of E&I methods via a tripartite scheme depicted in Figure 2 starting with abstract definitions and progressing toward the concrete. In so doing, we draw upon principles of inheritance and instantiation known from object oriented programming.

(a) Firstly, we define all possible parameter types that an editing method could use thus describing a *generic E&I method* in terms of its parameters. Parameter types have properties of their own that are also defined at this level.
(b) A *particular method's* parameters can then be defined by specifying each parameter's name and type. Additional information like the parameter's human-readable label and description are also provided. Parameters' properties need not be defined at this level, as they are inherited from the generic method's parameter types.
(c) Finally, the method is instantiated by giving its parameters concrete values. The resulting *method instance* can appear as a step in the process flow.

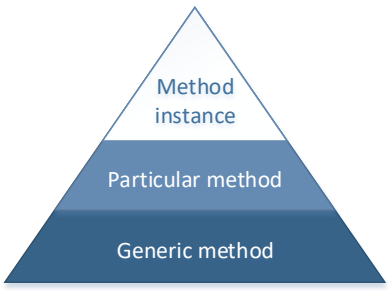

Figure 2: The layers in the modelling of E&I methods' parameters.

12. We note that it is possible for a process flow to contain several instances of the same method. An example would be imputing missing values by, first, searching for donors belonging to the same stratum as the recipient (e.g. in terms of sex and geographical region), then imputing the remaining missing values by allowing donors from any stratum. These process flow steps constitute instances of the same donor imputation method differing only in their parameter values.

B. Implementation at Statistics Finland

13. Statistics Finland has in recent years started experimenting on a larger scale with modelling metadata structures using the Resource Description Network (RDF) graph model and storing metadata in graph databases. The editing service's metadata are the most prominent example thus far. We have used RDF ontologies[1] to describe the generic method's parameter types and particular methods' parameters (the bottom two levels in Figure 2). The method instances' parameter values, in turn, are stored as RDF data. Both the ontologies and the data are stored in a graph database[2].

14. In order to accommodate Banff methods, the following parameter types have been described in the generic method's ontology.

(a) The *scalar parameter* is essentially a name-value pair. Scalars are used to set a method's options such as the threshold for outlier selection. The scalar parameter type has a property that allows the parameter's valid values to be constrained, e.g., to numeric values from a particular interval.
(b) The *vector parameter* accommodates several elements of the same type. Vector parameters are most commonly used to represent variable lists, e.g. the list of predictors in imputation models or in a distance function for donor imputation. The vector parameter may be weighted, meaning that each variable is ascribed a numeric value (used, e.g., to describe error localization weights).
(c) An *edit set* serves as a named container that contains a number of *edit rules*. Each edit rule is specified as an *expression* following a particular syntax (e.g. VTL) and may have properties of its own such as severity and error message.
(d) An *estimator set* contains a number of *estimators*. Each estimator, in turn, is specified via the estimation algorithm and target variable. Depending on the choice of algorithm, the estimator usually also references predictor variables (via a vector parameter) and has a number of options (scalar parameters).
(e) An *algorithm* defines a custom imputation algorithm (one not found on Banff's list of pre-defined algorithms).
(f) An *expression* is a piece of code in a particular programming language (e.g. SQL, SAS data-step) that can be used, e.g., when selecting a subset of the dataset to be edited.

15. Figure 7 in the appendix is a graphical representation of the estimator set–estimator–algorithm parameter structure. The corresponding structures are defined in the RDF ontology.

16. As mentioned, this allows us to describe a particular method's parameters by referring to the generic parameter types. Each parameter is given a name and a type and is additionally documented in human-readable form. Banff's outlier procedure, e.g., has the following parameters.

(a) A vector parameter (name: "var"), listing the variables for which outlier detection is to be performed.
(b) An optional vector parameter (name: "by"), listing variables that define the strata in which outlier detection is to be performed.
(c) A scalar parameter (name: "minobs"), that can take on non-negative integer values, specifying the minimum required number of observations in the data or stratum.
(d) A scalar parameter (name: "MII") that can take on positive decimal values, specifying the threshold for severe outlier selection.

(Banff's outlier procedure has other parameters that have been omitted here for brevity.)

[1] An RDF ontology is a formal way of describing structures in the RDF data like classes and properties. Its role is somewhat similar to that of an XML Schema that is used to describe the structure of XML documents.
[2] We have used the Apache Jena database that provides a web application point interface (API) to access and update the metadata.

17. When the particular method's parameters are defined in the ontology, the ontology (itself expressed as RDF data) can be queried via the graph database's API to return, e.g., the names and properties for all scalar-type parameters. This information can be used to dynamically generate the method's graphical user interface (GUI). Figure 8 in the Appendix is a screenshot of the GUI showing scalar parameters taken by Banff's outlier method. Dynamically generating the GUI minimizes the amount of coding required to add a new method to the editing service.

18. The Banff system covers the majority of E&I methods we wanted to provide to users of the editing service. In order to enable the editing service to run a particular Banff method, its parameters need to be modelled as described and the SAS-code executing the appropriate procedure needs to be "parametrized": re-written to include placeholders that will receive the parameter values.

IV. The if-then rule method

19. While Banff provides a wide range of methods, there was demand for the editing service to also be able to use a method that immediately and deterministically corrects values recognized as erroneous. This method is referred to as the if-then rule method in the GSDEMs document (UNECE 2015, p. 10) and is typically used to correct the thousand error using a correction rule of the form:

IF (conditions for thousand error) THEN (divide by 1000).

20. Because one of the editing service's goals is that it be implemented in a technology-neutral way, it was decided to adopt the Validation and Transformation Language (VTL 2016) as the native language for the declaration and storage of the correction rules.

21. The if-then rule method itself was implemented with SAS. Before the correction rules can be executed as part of a SAS data step, they must be accordingly translated. This required a translator to be developed that takes VTL-form rules and returns appropriate SAS data step code[3]. The translator was implemented as a standalone service with its own API enabling its re-use for other purposes[4].

22. Using VTL as the native language for the rules reduces Statistics Finland's long-term commitment to a particular technology, as the if-then rule method could in the future be re-implemented, e.g., in R. The transition would require that a VTL to R translator be developed while the content of the original correction rules would not need to be modified. If the correction rules were saved as SAS code, the transition to a new technology would be more difficult.

V. The E&I process flow

23. The previous sections described how parameters of E&I methods are modelled and stored in metadata repositories. In order to drive software, additional information is required, however, as the E&I process flow in general consists of more than one step (see UNECE 2015, Section 5 for a discussion). The order in which the process steps are to be executed is a vital part of the information that the editing service must take into account. A process flow can also contain controls that split the flow into branches or declare loops based on predefined criteria, further complicating the notion of the ordering of process steps.

24. Statistics Finland had already had experience with process management software that had been used to organize, automate, and schedule the execution of tasks. It had been decided that a new process

[3] The effort required to develop the translator was noticeably reduced because we were able to utilize the VTL parser Statistics Norway had developed and made available on GitHub.

[4] The translation service merely substitutes VTL function names with corresponding SAS function names and performs other minor alterations in order for the translated correction rule to follow SAS syntax. A comprehensive VTL to SAS translator would require substantially more effort but was not required for the task at hand.

management solution would be developed under the umbrella of the STIINA project that would be better tailored to Statistics Finland's needs and would allow for smoother integration with other services. Driving the execution of E&I actions is only one of the many uses the process management service will have at the statistical office.

25. Using the new solution, the user defines the process flow following BPMN 2.0 (OMG 2011) using a browser-based graphical user interface. In the case of an E&I process flow this boils down to positioning method instances as tasks of the process flow and possibly using controls to declare branches and/or loops. Figure 9 in the appendix is a screenshot of a trivial process flow in which a single review/selection task (Banff outlier) is followed by a single amendment task (Banff estimator).

VI. The editing service

26. The following sections describe the editing service's architecture and communication with metadata repositories as well as data flow, data structures, and reporting. We conclude with a discussion on the long-term reproducibility of E&I actions.

A. Architecture

27. The editing service's architecture follows the principles of Service Oriented Architecture mentioned in the introduction. It relies on the external process management service to determine the order of the tasks in the process flow to be executed. The translation of VTL rules to SAS has also been implemented as a standalone service enabling its future re-use. The editing service communicates with all metadata repositories and auxiliary services via the HTTP protocol, the single exception being the SQL databases (the staging area and reporting database) that are accessed via ODBC.

28. The editing service is accessed through its web API. A run of the editing process flow is initiated by submitting the following required inputs as a POST request (the arrow labelled 1a in Figure 3):

(a) the input dataset,
(b) the time period of the input dataset that will be edited,
(c) a reference to the input dataset's description,
(d) a reference to the process flow, and
(e) parameters pertaining to reporting.

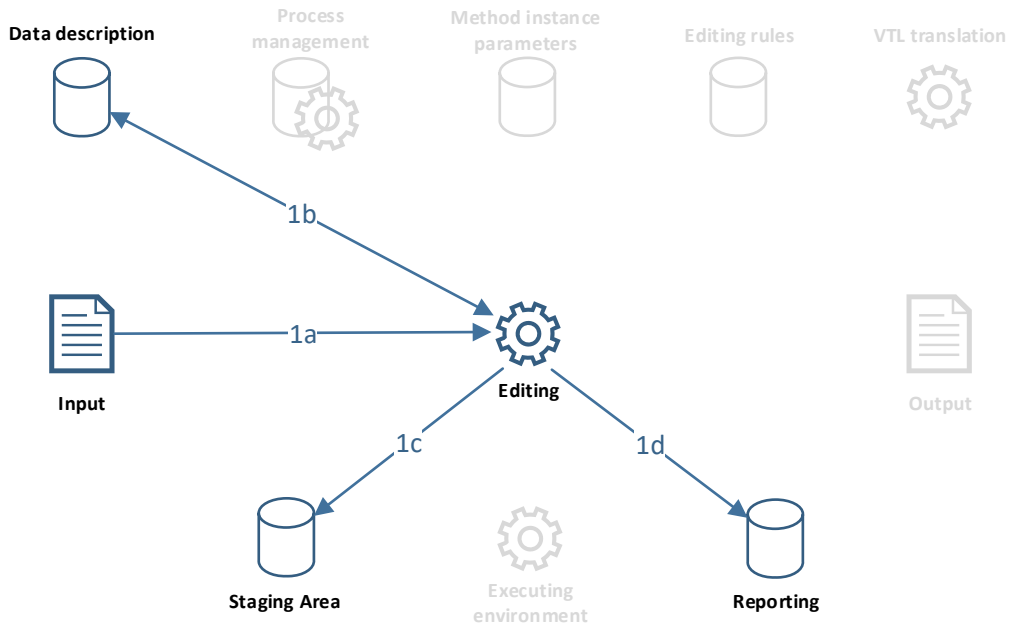

Figure 3: The editing service's initialization phase.

29. The editing service may also take the following optional inputs:
(a) audit trail with observations flagged for amendment,
(b) historical and other auxiliary datasets as part of the input dataset (see next section for discussion)
(c) time point for parameters.

30. Upon receiving the inputs, the editing service retrieves from the dataset description repository the datatypes of all the variables in the dataset to be edited (1b) and uses this information to initialize SQL tables in the staging area (1c). It also checks whether SQL-tables pertaining to the process flow in question already exist in the reporting database and creates them if they do not (1d).

31. The editing service run now enters the main phase and executes method instances one by one in the sequence defined by the process flow. The editing service queries the process management service and receives the ID of the method instance that is to be executed (the arrow labelled 2a in Figure 4). It then retrieves from the method instance repository the method instance's parameters (2b). If the method instance in question takes rules as parameters, the response will include the rules' IDs. The service uses the IDs to retrieve the rules' content expressed in VTL from the rules repository (2c) and submits the rules to the translation service to be translated to the language of the executing environment (SAS).

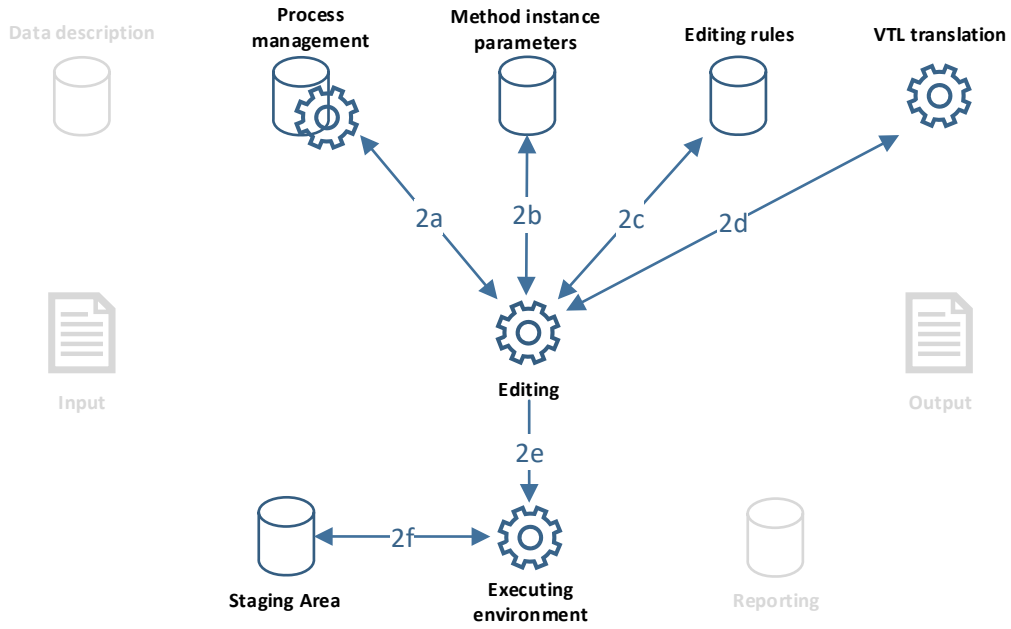

Figure 4: The editing service's execution phase.

32. Having retrieved all of the parameters from the metadata repositories, the editing service submits the method instance for execution to the executing environment. In the case of Banff, this boils down to passing the method instance's parameters to SAS (2e) and using them in the call to the appropriate procedure. Prior to the procedure call itself, the current version of the data to be edited and audit trail are retrieved from the staging area (2f). If needed, they are transformed to fulfill the particular method's technical requirements (e.g. Banff requires the input dataset be sorted according to stratification variables). After the method has executed, its outputs are used to update the datasets in the staging area.

33. The editing service then starts another execution cycle: it retrieves the ID of the next method instance to be executed (2a), retrieves its parameters from the metadata repositories and submits the method instance for execution. It does so until the process management service replies that the end of the process flow has been reached.

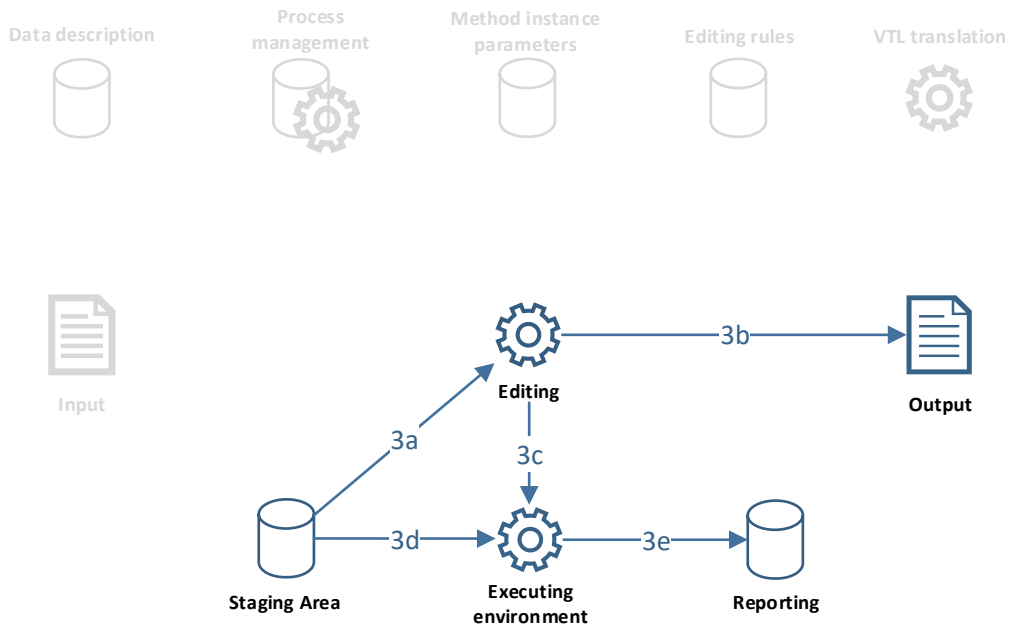

Figure 5: The editing service's termination phase.

34. The editing service then enters the termination phase, depicted in Figure 5. The current data are read from the staging area (3a), observations with no amendments are filtered out and the remainder is returned as the edited data (3b). The editing service also returns the audit trail with information on the editing actions performed in the execution of the process flow (3b). Next, a reporting procedure is called that reads the current audit trail from the staging area (3d), condenses its information into reports, and appends the reports to the appropriate tables in the reporting database (3e). Finally, the staging area is cleared.

B. Data flow and data structures

35. Whereas the previous section focused on the editing service's architecture and flow of metadata, the present section we will outline the editing service's data flow and data structures. We note that the distinction between metadata (parameters) and data that we make here is a technical one: we use the term data to describe any information that resides in datasets in the executing environment and in SQL tables. The information contained in the audit trail dataset, e.g., flags that selection methods "pass" to amendment methods to inform them that certain data points have been selected for amendment, would be considered referential *meta*data rather than data according to GSDEMs (UNECE 2015).

36. The editing service's datasets follow one of two data structures.
   (a) The *wide* data structure is the common one where a row corresponds to an observation. One or more columns record the observation's identification information while the remaining columns contain values describing the observation's characteristics. We note that all datasets the editing service uses must contain a *timePeriod* column identifying the data collection cycle.
   (b) The *long* data structure, on the other hand, contains an additional identifier column (called *variableName*) that identifies the variable (characteristic) in question. One row of a dataset in long form therefore corresponds to a particular *data point* of the dataset in wide form.

37. The long data structure is used to record detailed information on data points that are the subject of editing actions. When, e.g., a selection method flags a data point for amendment, this fact is recorded along with information on the method instance used. When, conversely, a data point passes through a method instance without being flagged or amended, it is simply left out of the audit trail dataset. The audit trail dataset includes the following columns.
   (c) Identifier variables identifying the observation, including the mandatory *timePeriod* identifier.
   (d) *VariableName* that identifies the variable within the observation that is the object of the editing action.

(e) *Value* and *severity* are numeric columns whose meaning depends on the purpose of the method in question.
- For a "pure" review method (one that does not also perform selection) *value* contains the data point's value from the dataset to be edited while *severity* indicates the degree of implausibility associated with that value with higher values of severity indicating greater degrees of implausibility.
- For selection methods, *value* contains the data point's value from the dataset to be edited while *severity* indicates whether the data point has been flagged for amendment[5] (in which case severity=1).
- For amendment methods, *value* contains the new (amended) value while *severity* is empty.

(f) *Timestamp* gives the date and time of the editing action.
(g) The *method* column identifies the E&I method used for the editing action in question.
(h) The *methodInstance* column contains the method instance identifier that serves as a link between the audit trail dataset and the method parameters in the metadata repositories. No additional information on the parameters needs to be included in the audit trail. If one wishes to examine an editing action, one can obtain the parameter values the method was called with from the metadata repositories using the *methodInstance* and *timestamp* information.
(i) Since the process flow can possibly contain loops, it is possible that the same method instance be executed several times within a run of the process flow. The *methodInstanceExecution* column is used to identify executions of method instances.
(j) The *elaboration* column identifies a "standalone parameter" within the method instance that was used for the editing action in question. This might be the correction rule's identifier in case of the if-then rule method or the estimator identifier in case of model-based imputation. *Elaboration* is empty if the method in question uses no such standalone parameters.

38. A simplified portray of the editing service's dataflow is given in Figure 6. The editing service receives the input data in wide form. This dataset contains the data to be edited, but might also contain auxiliary data, such as historical data from previous collection cycles that can be utilized by some methods (e.g. model-based imputation). The input dataset is split into the data to be edited and historical data based on the time period parameter that must be provided in the editing service call. The input data could also contain other micro-level auxiliary data such as donor pools that would similarly need to be split off using an indicator variable.

39. The editing service can take an input audit trail in long form. When such an input is given, its content is transferred to the corresponding SQL table in the staging area. When no input audit trail is provided, a blank table is initiated in the staging area.

40. The editing service run then enters the main phase executing method instances. Each method has an associated transformation routine that reads the data to be edited and audit trail from the staging area and transforms them to fulfill the method's technical requirements. The *input status* dataset produced by the transformation routine is similar in structure to the audit trail but only contains the most recent relevant information on each data point. This means, e.g., that an amendment method's input status will contain all data points selected for amendment except those data points that were already amended by preceding method instances performing amendment.

41. Once the method has been run, its outputs are back transformed to be compatible with the datasets in the staging area. The data to be edited are updated on the basis of the amendments that the method instance performed. Corresponding rows of the output status dataset are, in turn, appended to the audit trail dataset in the staging area. We note that whereas the dataset to be edited is continuously updated during the process run, existing rows of the audit trail are never modified – the audit trail is always appended to, never updated.

[5] Banff's outlier method also allows a data point to be flagged "field to be excluded" (FTE). This is a milder form of outlier that will not itself be amended. In case donor estimation is subsequently used for amendment, however, the FTE flag will prevent a mild outlier to be used as a donor. We have used severity=0.5 to denote FTE data points.

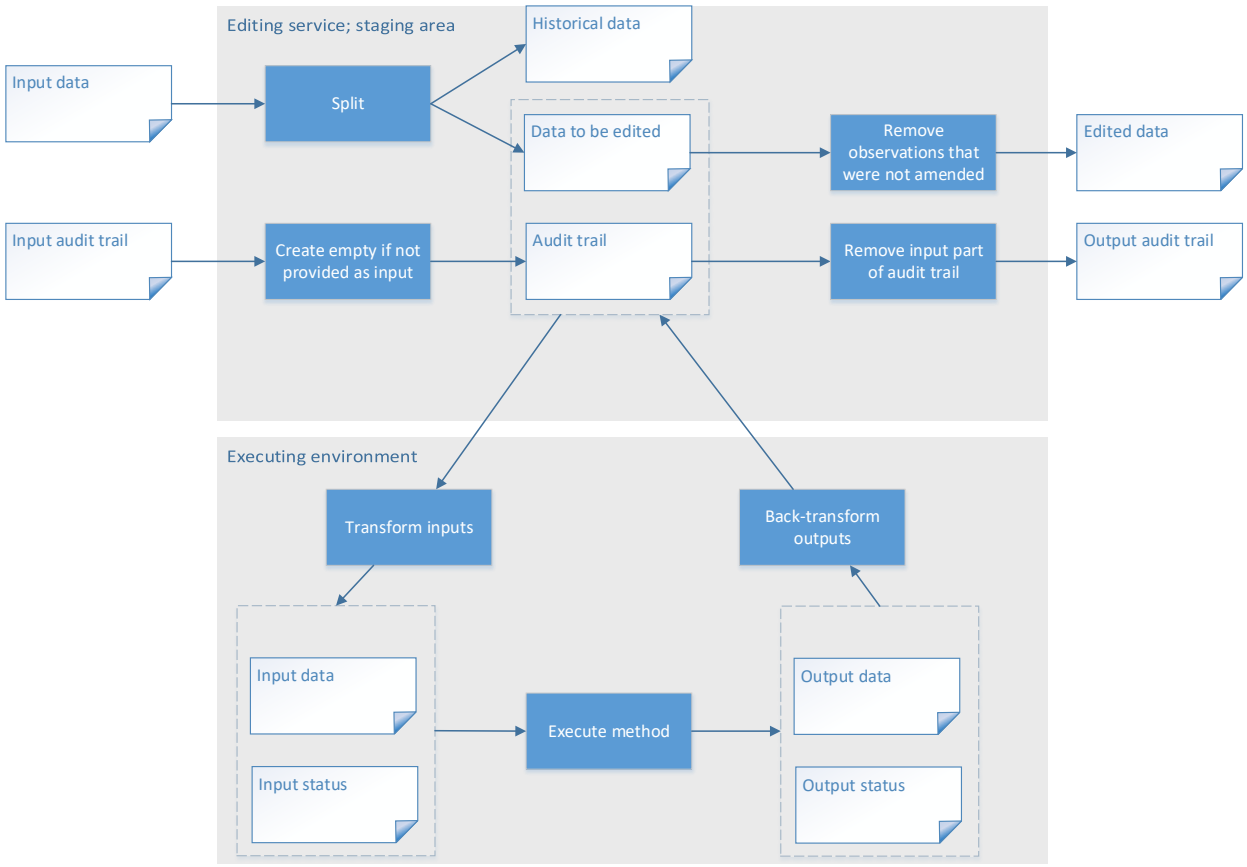

Figure 6: The editing service's data flow diagram.

42. Once the process flow run has finished, the editing service 1) filters out observations that were not amended during the run and returns the remainder as the edited data and 2) returns the audit trail from which it removes the part that might have been provided as input. The editing service also calls a reporting procedure that produces reports on the basis of the audit trail (reporting is described in more detail in the next section).

43. We note that the method's purpose determines its inputs and outputs in terms of data as has already been discussed in the GSDEMs document (UNECE 2015, p. 9-10). Methods performing review and/or selection take the data to be edited but no status dataset and return only a status dataset with flagged data points, but no amended data. Amendment methods, conversely, take both the dataset to be edited and an input status and return both the edited dataset and an output status.

44. The if-then rule method is somewhat of an exception in that it also performs review/selection and does therefore not require an input status. Its output status is also exceptional in that each correction is documented by two rows: the first row gives the old value flagged with *severity*=1 and the second row gives the corrected value.

C. Reporting

45. While the audit trail contains detailed information on the process flow run, it would be difficult to interpret its raw figures to answer any pertinent questions about the run such as "how many observations were flagged and amended in a subgroup of interest?" In order to enable such interpretations, the editing service produces standardized aggregated reports. Quality indicators such as the imputation rate can easily be calculated on the basis of the reports. The editing service produces the following SQL-tables in the reporting database for each process flow.

(a) The *frequency* report contains the information on the number of flagged and amended data points and observations according to method instance execution and subgroups defined by reporting variables.

(b) The *variable frequency* report contains the same data point frequencies further structured according to target variable.
(c) The *impact* report gives the information on the impact the editing has had on statistics of choice (e.g. mean, median, standard deviation). The impact report, also, is structured according to method instance execution and subgroups. Statistics of choice are calculated before and after each method instance execution.
(d) The *metrics* report gives information on how the process run such as the duration of each method instance execution, possible errors, warnings etc. It is the only type of report that is not calculated on the basis of the final audit trail but is rather produced as each method instance is run.

46. The report tables are created in the reporting database when a process flow is run for the first time. In each subsequent run, rows are appended to the existing tables. The idea is to accumulate information to the same tables facilitating easy comparisons between runs (data collection cycles). If, e.g., the number of flagged observations in a particular subgroup were to substantially increase as compared to previous collection cycles, we would be prompted to further investigate whether and how the input data have changed.

D. A note on reproducibility

47. Even though it can be said to provide transparency and traceability of E&I actions, the detailed information in the audit trail does not assure the reproducibility of results. As the content or structure of the data to be edited changes with time, the person maintaining the metadata will need to modify the method instances' parameter values to accommodate for the changes. Similarly, modifications might need to be made to the process flow reordering method instances or introducing new ones. If the maintenance of the processing metadata involves discarding (overwriting) old values, results of editing actions prior to the change cannot be replicated even if the original data are still available.

48. In order to ensure reproducibility, the editing service's metadata repositories should follow the so-called immutable database design, never overwriting old values. Instead, as a parameter value is changed, the old value is kept and assigned an expiration time. This allows us to retrieve from the repository the version of the parameter values that were valid at a particular point in time. In fact, if we want to ensure the reproducibility of results, *all* of the editing service's inputs should be versioned this way, including possible auxiliary data.

49. Because of time constraints, we have not been able to structure the editing service's metadata repositories according to immutable database design. We have, however, demonstrated via a proof of concept how this could be achieved and will continue pursuing reproducibility of E&I actions as a goal.

VII. Discussion and further work

50. At the time of writing, Statistics Finland's editing service is still actively under development and has only seen usage in testing. It is still missing a number of features, e.g., only a handful of Banff methods have so far been implemented. To improve user-friendliness, thorough parameter and data consistency checks will need to be introduced allowing the service to return intelligible error messages.

51. During the development project's reviews, potential users have expressed a concern that the editing service's many layers reduce the transparency of E&I actions. Correction rules are, e.g., written in VTL by the user, but their translation and execution occurs in a "black box". The editing service's logging will need to be improved in order to address these concerns.

52. All in all, however, we consider the project implementing the editing service to have been very successful. We have shown how even complex functionalities like editing and imputation can be implemented in a generic and fully metadata-driven way. A similar architecture can be utilized for future metadata-driven services to be developed at Statistics Finland, e.g., ones implementing data aggregation or computation of derived variables.

53. We believe that the way the service's datasets are structured and how the methods' parameters are modelled is generic enough to allow new E&I methods to be added to the editing service's method library in the future. Even though all the editing service's currently supported methods are executed in SAS, the service's architecture allows methods to be added that use, e.g., R or Python as the executing environment. Methods with different executing environments can be used in the same process flow.

54. Finally, a pervasive theme throughout the project has been standardization. The structure of the datasets flowing into and out of the editing service have been standardized. A scheme has been proposed according to which the parameters of new methods can be modelled. The editing service automatically produces reports with a static structure and appends them to existing tables as data for new collection cycles are processed. The reports' static structure allows further services to be developed that compute quality indicators on the basis of the reports. The accumulation of standard reports over time, on the other hand, allows for in-depth analyses of how E&I is performed at the statistical office informing further development in the field.

References

Fellegi, I.P. and Holt, D. (1976). A systematic approach to automatic edit and imputation. *Journal of the American Statistical Association* (71), 17-35.

OMG (2011). *Business Process Model and Notation (BPMN)* Version 2.0. OMG Final Adopted Specification. Object Management Group. Available at: https://www.omg.org/spec/BPMN/2.0/PDF (Accessed 7 June 2018).

Pannekoek, J. and Zhang, L. (2012). *On the general flow of editing*. Oslo: UNECE Conference of European statisticians.

Platinovsek, R. (2017). *An information model for a metadata-driven editing and imputation system*. The Hague: UNECE Conference of European statisticians.

Statistics Canada. (2012). *Banff Processor User Guide*. Ottawa: Statistics Canada.

Statistics Canada. (2014). *Functional description of the Banff system for edit and imputation*, Version 2.06. Ottawa: Statistics Canada.

UNECE. (2013a). *Generic Statistical Business Process Model*. Version 5.0 UNECE: Conference of European statisticians.
Available at: https://statswiki.unece.org/display/GSBPM/GSBPM+v5.0 (Accessed 7 June 2018).

UNECE. (2013b). *Generic Statistical Information Model (GSIM): Specification*. Version 1.1 UNECE: Conference of European statisticians.
Available at: https://statswiki.unece.org/display/gsim/GSIM+Specification (Accessed 7 June 2018).

UNECE. (2015). *Generic Statistical Data Editing Models*. Version 1.0 UNECE: Conference of European statisticians. Available at: https://statswiki.unece.org/display/sde/GSDEMs (Accessed 7 June 2018).

VTL, 2016. *Validation and Transformation Language version 1.1, Part 2 - Reference Manual*, SDMX Technical Working Group: VTL Task Force.

Appendix

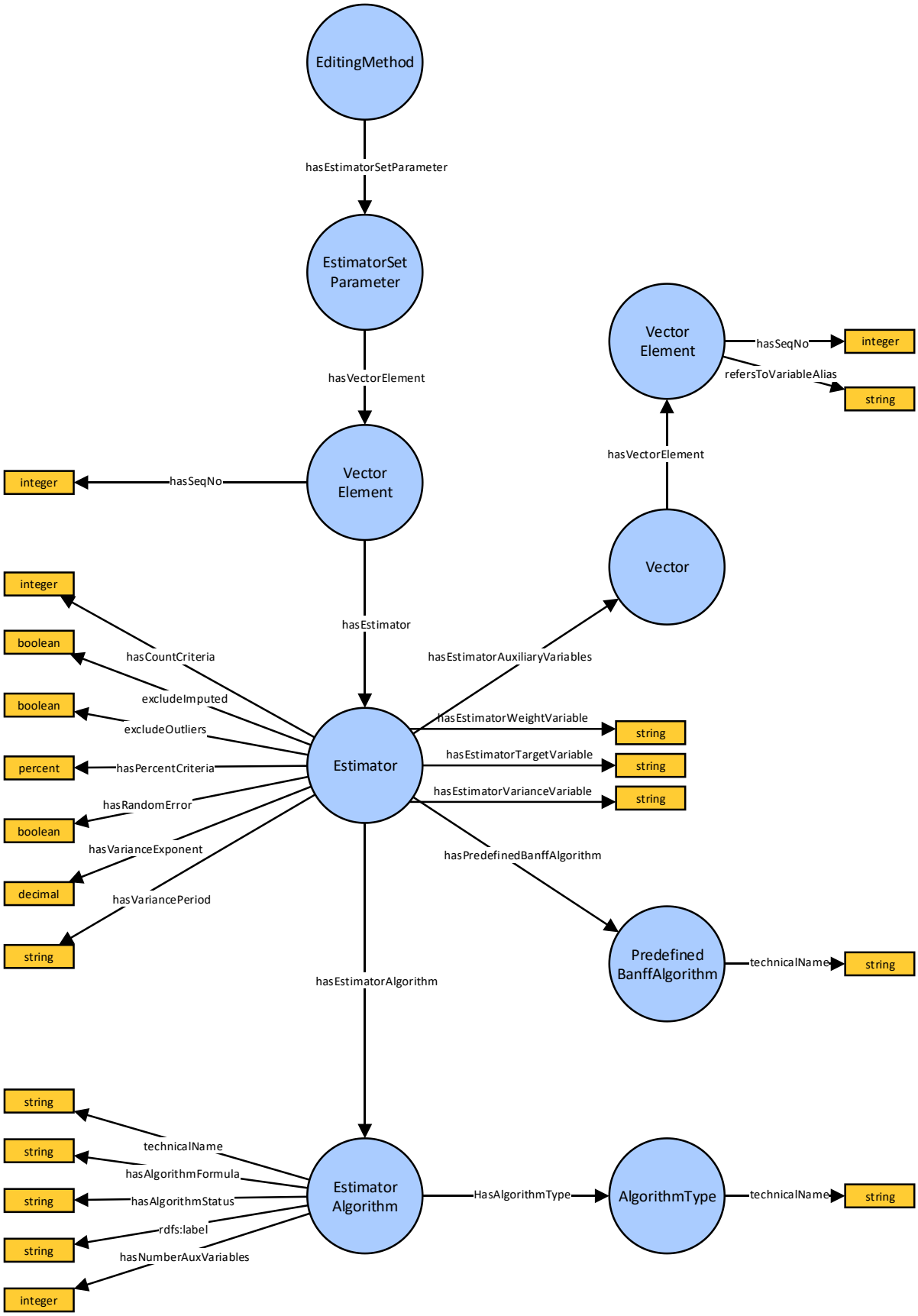

Figure 7: Graphical representation of the estimator set–estimator–algorithm parameter structure.

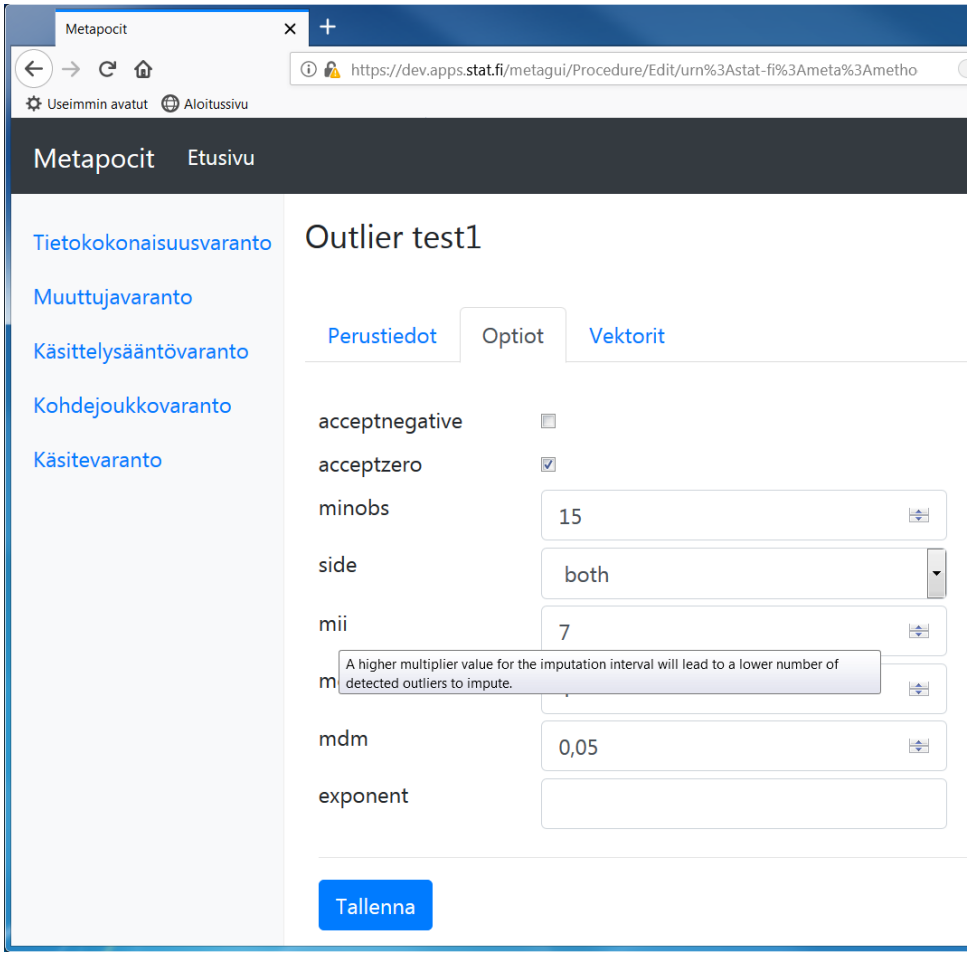

Figure 8: A dynamically generated web-form with scalar parameters of the Banff Outlier Method. The flyover text, too, is read from the method's ontology and displayed when the user places the mouse cursor over the parameter name.

Figure 9: A trivial E&I process flow in Statistics Finland's process management GUI.