3.4 Data-Centric workflow

One of the most important activities in a S-DWH environment is represented by data integration of different and heterogeneous sources. The process of extract, transform, and load data from different heterogeneous statistical data sources into a single schema so that data become compatible with each other and can be processed, compared, queried or analysed regardless of the original data structure and semantics.

This sub-chapter deals with the integration of processes and all the elements needed for the data process phase. We structure the process phase in a S-DWH as a data-centric workflow. The data-centric WF is characterized by frequently modified processes, process re-use or adjustments. As example, in the case of administrative data source not under the direct control of statisticians; the sources structure or content may change each supply which implies adaptation of the data integration processes or completely rewrite the procedures.

In order to efficiently organize the WF\(^1\) with the aim to support the production processes and improve quality, it is necessary to connect several entities such as the source variables and the related documentation. It is also important to gather and record the versions of any entity in order to fully document the process and guarantee its quality, reliability, replicability and reusability. A systematic collection of all the tested attempts could also contribute to the production efficiency because the researcher’s team would be able to examine all past discarded hypotheses.

All previous consideration bring us to the following item functionality needs:

- **design and management of a data centric workflow.** It allows designing, modifying and executing the main phases, sub-processes and elementary activities which constitute the statistical production process;

- **activities and processes schedule.** The activities, the remote processes and the procedures can be run by a scheduler in an automatic way. This is particularly useful when one deals with huge amounts of data. The scheduler’s purpose is to translate the workflow design into a sequence of activities to be submitted to the distributed processing nodes. This sequence has to satisfy priority constraints planned during the design phase;

- **local and remote services call.** Each elementary activity can be either a native procedure (e.g. a SAS procedure, a PL/SQL program or an R procedure) or an external service, such as a web service encapsulating a high-level domain service (i.e. BANFF) that can be invoked from the platform. It is necessary to provide some mechanism of sharing information between systems;

- **integration of statistical abstractions.** A statistical production process has its own rules, constraints, methodologies and paradigms. The aim of the statistical abstraction layer is to supply a set of abstractions that make the researcher’s work flexible, independent of technical details and more focused on research objectives. Among the possible abstractions there could be:
  - **meta-parameters:** the use of global parameters reduces the need to modify the scripts and variables necessary for other systems to operate correctly;
  - **partitioning or filtering units:** each type of record (unit) has its own processing path in the WF. The value of some variable could be used to filter units to the next processing step;
  - **sampling test:** when the amount of data is very large, it is useful to test some hypothesis or

---

programs on a subset of data in order to avoid loss of time and to early discover weak hypotheses;

- **rule checker**: a tool for finding inconsistencies in a formally defined set of rules and to manage efficiently semantic and definitional changes in sources;

- **documentation management and versioning. It is possible to associate one or more documents and metadata** to each WF element and, at any time, recall previous versions of the WF and all the elements connected.

- **metadata** module implements a decoupling approach in data mapping. This type of abstraction introduces a new layer between data sources and statistical variables so that a semantic change in one administrative source does not affect statistical sub-processes that depend on the related statistical variables;

- **rules** module allow the researcher to write the consistency plan, check possible contradictions in the edits set, run the plan, log error and warnings and produce reports. Moreover, this module assists the researcher in the activity of modifying an existing check plan in case some variables are introduced or deleted;

- **parameters** module is used to implement a basic form of parametric changes in all of the components of the WF. It can be thought as similar to a dashboard through which modifying thresholds, setting parameters, choosing elaborative units, switching on and off options, etc. For instance, suppose one parameter is shared by many sub-processes: a change in this value has an impact on all the sub-processes containing that parameter. The parameter is a placeholder that at runtime is set to the actual value (e.g. some sub-process can possibly change the parameters’ value during processing);

- **processes** module provides information on actual state of active elaborations. It is possible to view the scheduled sequence of sub-processes and to recall the log of previous ones;

- **procedure editor** module is the development environment needed to create procedures or modify existing code. Such a module should support at least one statistical language (SAS, R) and one data manipulation language (PL/SQL). New languages can be added to this system in a modular and incremental way. The editor integrates a versioning system in order to restore a previous version of a procedure, document code changes and to monitor the improvements of the implemented functions;

- **the micro-editing component** is used in manual and interactive micro data editing activities. It can be a useful tool for statisticians to analyze some sample of micro-data.
3.4.1 Example of modularity

This paragraph in more depth focuses on the Process phase of the statistical production. Looking at the Process phase in more detail, there are sub-processes in it. These elementary tasks are the finest-grained elements of the GSBPM. We will try to sub-divide the sub-processes into elementary tasks in order to create a conceptual layer closer to the IT infrastructure. With this aim we will focus on “Review, validate, edit” and we will describe a possible generic sub-task implementation in the next lines.

Let's take a sample of five statistical units (represented in the following diagram by three triangles and two circles) each containing the values from three variables (V1, V2 and V3) which have to be edited (checked and corrected). Every elementary task has to edit a sub-group of variables. Therefore, a unit entering a task is processed and leaves the task with all that task's variables edited.

We will consider a workflow composed of 6 activities (tasks): S (starting), F (finishing), and S1, S2, S3, S4 (editing data and activities). Suppose also each type of unit needs a different activity path, where triangle shaped units need more articulated treatment on variables V1 and V2. For this purpose a “filter” element F is introduced (the diamond in the diagram), which diverts each unit to the correct part of the workflow. It is important to note that only V1 and V2 are processed differently because in task S4 two branches rejoin.
During the workflow, all the variables are inspected task by task and, when necessary, transformed into a coherent state. Therefore each task contributes to the set of coherent variables. Note that every path in the workflow meets the same set of variables. This incremental approach ensures that at the end of the workflow every unit has its variables edited. The table below shows some interesting attributes of the tasks.

<table>
<thead>
<tr>
<th>Task</th>
<th>Input</th>
<th>Output</th>
<th>Purpose</th>
<th>Module</th>
<th>Data source</th>
<th>Data target</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>All units</td>
<td>All units</td>
<td>Dummy task</td>
<td>-</td>
<td>TAB_L_I_START</td>
<td>TAB_L_II_TARGET</td>
</tr>
<tr>
<td>S₁</td>
<td>Circle units</td>
<td>Circle units (V₁, V₂ corrected)</td>
<td>Edit and correct V₁ and V₂</td>
<td>EC_V₁(CU, P₁)</td>
<td>EC_V₂(CU, P₂)</td>
<td>TAB_L_II_TARGET</td>
</tr>
<tr>
<td>S₂</td>
<td>Triangle units</td>
<td>Triangle units (V₁ corrected)</td>
<td>Edit and correct V₁</td>
<td>EC_V₁(TU, P₁₁)</td>
<td></td>
<td>TAB_L_II_TARGET</td>
</tr>
<tr>
<td>S₃</td>
<td>Triangle units (V₁ corrected)</td>
<td>Triangle units (V₁, V₂ corrected)</td>
<td>Edit and correct V₂</td>
<td>EC_V₂(TU, P₂₂)</td>
<td></td>
<td>TAB_L_II_TARGET</td>
</tr>
<tr>
<td>S₄</td>
<td>All units (V₁, V₂ corrected)</td>
<td>All units (all variables corrected)</td>
<td>Edit and correct V₃</td>
<td>EC_V₃(U, P₃)</td>
<td></td>
<td>TAB_L_II_TARGET</td>
</tr>
<tr>
<td>F</td>
<td>All units</td>
<td>All units</td>
<td>Dummy task</td>
<td>-</td>
<td>TAB_L_II_TARGET</td>
<td>TAB_L_III_FINAL</td>
</tr>
</tbody>
</table>

**Figure 2 - Review, Validate and Edit on Process Phase**

During the workflow, all the variables are inspected task by task and, when necessary, transformed into a coherent state. Therefore each task contributes to the set of coherent variables. Note that every path in the workflow meets the same set of variables. This incremental approach ensures that at the end of the workflow every unit has its variables edited. The table below shows some interesting attributes of the tasks.
The columns in the table above provide useful elements for the building and definition of modular objects. These objects could be employed in an applicative framework where data structures and interfaces are shared in a common infrastructure. The task column identifies the sub-activities in the workflow: the subscript, when present, corresponds to different sub-activities. Input and output columns identify the statistical information units that must be processed and produced respectively by each sub-activity. A simple textual description of the responsibility of each sub-activity or task is given in the purpose column. The module column shows the function needed to fulfill the purpose. As in the table above, we could label each module with a prefix, representing a specific sub-process EC function (Edit and Correct), and a suffix indicating the variable to work with. The first parameter in the function indicates the unit to treat (CU stands for circle unit, TU for triangle unit), and the second parameter indicates the procedure (i.e. a threshold, a constant, a software component).

Structuring modules in such a way could enable the reuse of components. The example in the table above shows the activity S1 as a combination of EC_V1 and EC_V2 where EC_V1 is used by S1 and also S2 and EC_V2 is used by S1 and also S3. Moreover, because the work on each variable is similar, single function could be considered like a skeleton containing a modular system in order to reduce building time and maximize re-usability. Lastly, the data source and target columns indicate references to data structures necessary to manage each step of the activity in the workflow.

3.4.2 Towards a modular approach
There are many software models and approaches available to build modular flows between layers. S-DWH’s layered architecture itself provides possibility to use different platforms and software in separate layers or to re-use components already available. In addition, different software can be used inside the same layer to build up one particular flow. The problems arise when we try to use these different modules and different data formats together.

One of the approaches is CORE services. They are used to move data between S-DWH layers and also inside the layers between different sub-tasks, then it is easier to use software provided by statistical community or re-use self-developed components to build flows for different purposes.

CORE services are based on SDMX standards and use main general conception of messages and processes. Its feasibility to use within statistical system was proved under ESSnet CORE. Note that CORE is not a kind of software but only a set of methods and approaches. Generally CORE (COmmon Reference Environment) is an environment supporting the definition of statistical processes and their automated execution. CORE processes are designed in a standard way, starting from available services. Specifically, process definition is provided in terms of abstract statistical services that can be mapped to specific IT tools. CORE goes in the direction of fostering the sharing of tools among NSIs. Indeed, a tool developed by a specific NSI can be wrapped according to CORE principles, and thus easily integrated within a statistical process of another NSI. Moreover, having a single environment for the execution of entire statistical processes, it provides a high level of automation and a complete reproducibility of processes execution.

The main principles underlying CORE design are:
a) **Platform Independence.** NSIs use various platforms (e.g. hardware, operating systems, database management systems, statistical software, etc.), hence architecture is bound to fail if it endeavours to impose standards at a technical level. Moreover, platform independence allows to model statistical processes at a “conceptual level”, so that they do not need to be modified when the implementation of a service changes.

b) **Service Orientation.** The vision is that the production of statistics takes place through services calling other services. Hence services are the modular building blocks of the architecture. By having clear communication interfaces, services implement principles of modern software engineering like encapsulation and modularity.

c) **Layered Approach.** According to this principle, some services are rich and are positioned at the top of the statistical process, so, for instance a publishing service requires the output of all sorts of services positioned earlier in the statistical process, such as collecting data and storing information. The ambition of this model is to bridge the whole range of layers from collection to publication by describing all layers in terms of services delivered to a higher layer, in such a way that each layer is dependent only on the first lower layer.

In a general sense, an integration API permits to wrap a tool in order to make it CORE-complaint, i.e. a CORE executable service. CORE service is indeed composed by an inner part, which is the tool to be wrapped, and by input and output integration APIs. Such APIs transform from/to CORE model into the tool specific format.

Basically, the integration API consists of a set of transformation components. Each transformation component corresponds to a specific data format and the principal elements of their design are specific mapping files, description files and transform operations.

### 3.4.3 Possible work flow scenarios
Layered architecture, modular tools and warehouse-based variables is a powerful combination that can be used for different scenarios. Here are some examples of workflows that S-DWH supports.

#### 3.4.3.1 Scenario 1: full linear end-to-end workflow
To publish data in access layer, raw data need to be collected into raw database in source layer, then extracted into integration layer for processing, then loaded into warehouse in interpretation layer. After that, someone can calculate statistics or make an analysis and publish it in access layer.
3.4.3.2 Scenario 2: Monitoring collection
Sometime it is necessary to monitor collection process and analyse the raw data during the collection. Then the raw data is extracted from the collection raw database, processed in integration layer so that the data can be easily analysed with specific tools and be used for operational activities, or loaded to interpretation layer, where it can be freely analysed. This process is repeated as often as needed – for example, once a day, once a week or hourly.
3.4.3.3 Scenario 3: Evaluating new data source
When we receive a dataset from a new data source, it should be evaluated by statisticians. Dataset is loaded by the integration layer from the source to the interpretation layer, where statisticians can make their source-evaluation or, due to any changes on the administrative regulations, define new variables or new process-up-date for existents production process. From the technical perspective, this workflow is same as described in scenario 2. It is interesting to note that this update must be included in the coherent S-DWH by proper metadata.

3.4.3.4 Scenario 4: Re-using data for new standard output
Statisticians can analyse data already prepared in integration layer, compile new products and load them to access layer. If S-DWH is built correctly and correct metadata is provided, then compiling new products using already collected and prepared data should be easy and preferred way of working.

![Figure 5 - Data Re-Use for New Output](image)

3.4.3.5 Scenario 5: re-using data for complex custom query
This is variation from scenario 4, where instead of generating new standard output from data warehouse, statistician can make ad-hoc analysis by using data that is already collected and prepared in warehouse and prepare custom query for customer.
Figure 6 - Data Re-Use for Custom Query