DE GRUYTER
OPEN

Journal of Official Statistics, Vol. 34, No. 4, 2018, pp. 835–862, http://dx.doi.org/10.2478/JOS-2018-0042

Efficiency and Agility for a Modern Solution of Deterministic Multiple Source Prioritization and Validation Tasks

Annalisa Cesaro[1] *and Leonardo Tininini*[1]

This article focuses on a multiple source prioritization and validation service. We describe a modern rule-based, loosely coupled solution. We follow generalization, efficiency and agility principles in application design. We show benefits and stumbling blocks in micro-service architectural style and in rule-based solutions, where even the selection task is solved through selection rules, which encapsulate the calls to Entity Services, allowing access to input-sources. We allowing the rule-based service efficiency and further local and remote input data selection scenarios for the validation Statistical Service. In particular, data virtualization technologies enable architects to use remote sourcing and further increases agility in data selection issues. Through a wide number of experimental results, we show the necessary level of attention in process implementation, data architectures and resource usage. Agility and efficiency emerge as drivers which possibly sustain the Modernization flexibility impetus. In fact, flexible services may potentially serve multiple scenarios and domains.

Key words: Rule engine for validation and prioritization; highly-performant data management; efficient data parallelism; data virtualization; agile culture.

1. Background

Over the last decade, National Statistical Offices (NSOs) have been investing heavily in new approaches to improve and make more flexible their data supply chains. Modernization efforts in official statistics require the reuse and sharing of methods, components, processes and data repositories.

1.1. Sharing and Reuse Needs: The SOA Impetus

There is a growing trend to introduce Service-Oriented Architectures (SOAs) in official statistics due to their promise of cost efficiency, agility, adaptability and legacy leverage (O'Brien et al. 2008). SOA is based on sharable independent services, which should be (i) efficient, thus being without resource waste and possibly usable for small and large data sets; (ii) agile, thus easily managing constantly evolving scenarios; and (iii) generalized, thus applicable in cross-cutting domains. It is worth noting that several SOA architectural styles exist (Quensel-von Kalben 2017a), for example (i) point to point integration;

[1] Italian National Institute of Statistics (Istat), via Balbo 16, 00184 Roma, Italy. Emails: cesaro@istat.it and Tininini@istat.it

Acknowledgments: Monica Scannapieco for Modernisation discussion, Monica Consalvi and Francesca Alonzi for SBR validation rules, Laura Maglione for enterprise metadata knowledge and legacy validation rules, Marco Passacantilli for java web app refactoring/development.

(ii) platform integration, based on an Enterprise Service Bus, which interconnects mutually interacting components through events or messages management; and (iii) lightweight basic integration, by using autonomous fine-grained micro-services, which are unaware of their position in the process chain/control flow (Fowler 2014; Namiot and Sneps-Sneppe 2014). In the literature, there is still a lack of consensus on what micro-services actually are (Dragoni et al. 2017). A micro-service should maintain focus on providing a single business capability, moreover each micro-service should be operationally independent from others. In such architecture, the only form of communication among services is through their interfaces. Many migration patterns towards a SOA exist in the literature, as outlined by Razavian and Lago (2015) and Khadka et al. (2012).

As highlighted by Quensel-von Kalben (2017b), Enterprise Architecture is a fundamental driver in official statistics modernization actions. The Business Architecture of reference for official statistics is the Generic Statistical Business Process Model (GSBPM), which identifies business functionalities that need to be supported by IT systems. The processes and sub-processes of the GSBPM may rely on different information models (GSBPM v5.0 2017). IT systems, fulfilling the business needs, define the application architecture, which is recommended to be migrated towards a SOA. When SOA is used, the Common Statistical Production Architecture (CSPA) should be taken into account (ESSnet 2015). CSPA states the main design principles to be followed when assessing the design of the business, information and application architectural layers. Briefly, CSPA-compliant services rely on the matching of the service functionalities with one or more GSBPM activities, as well as on non-functional requirements, such as performance (i.e., resource utilization, time behavior and capacity), scalability, security, and language. The abstract information model of reference relies on the Generic Statistical Information Model (GSIM), which could be mapped onto more specialized information models that are in use in official statistics. When used, GSIM enables harmonization in service definition and greater decoupling between statistical domain experts and Information Technology (IT) ones. Before CSPA, Eurostat organized CORE (ESSnet Core Project 2011), which designed a platform to orchestrate GSBPM-compliant Statistical Services: it promoted the idea that the data model – that is, the inputs and outputs of the services – might be described through the GSIM. However, service-sharing across NSOs is a difficult activity and remains a work in progress, as outlined by Quensel-von Kalben (2017a).

1.2. Data Virtualization for Avoiding Silos: A Focus on Performance

Dealing with different types of data sources is also a challenging issue in modern IT systems. Official statistics require many different sources to be integrated in the statistical process. In particular, integration may involve (i) large and unstructured data collections, (ii) performing classical relational, and optimized data management, and for official statistics, either (iii) Resource Description Framework (RDF) data, when standardized and linked open data are used, or (iv) Statistical Data and Metadata eXchange (SDMX) data.

Data virtualization is a relatively new approach to data selection and integration (Pullokkaran 2013) that avoids physically moving data into a single integrated environment and reduces the risk of integration silos (Pullokkaran 2013; Alagiannis

et al. 2012; Karpathiotakis et al. 2015). Several data virtualization patterns exist: (i) data may be passed directly to an execution engine for query processing and then discarded; (ii) data may be cached in memory for subsequent processing and then discarded; or (iii) data may be temporarily written to disk for prompt subsequent processing and then discarded (Idreos et al. 2011; Cheng and Rusu 2015). Multi-source data integration and/or data cleansing and transformation might hence be defined in a logical layer and then applied to data as they are retrieved from the data sources while generating reports (Pullokkaran 2013; Krawatzeck et al. 2015). Databases may be built by launching queries, instead of building databases for launching queries (Karpathiotakis et al. 2015). Such techniques are used nowadays in, for example, the agile Business Intelligence (BI) context (Stodder 2013; Van Der Lans 2013).

Performance is often evaluated in terms of execution time (Karpathiotakis et al. 2015; Alagiannis et al. 2012; Tian et al. 2017) and parallelism can be a rewarding technique in virtual loading (Cheng and Rusu 2015). For an increased performance, service replication in distributed systems may be tackled (Osrael et al. 2006; Chen et al. 2014; Mohamed 2016; Xie et al. 2017). Server resource consumption has to be considered (Xavier et al. 2013) as well. Conversely, when data locality is ensured, data replication and data consistency issues have to be taken into account (Montoya et al. 2017), as well as storage and energy consumption (Milani and Navimipour 2016). The same architectural issues are common in official statistics, where replicated services versus shared services are evaluated in the European Statistical System network (ESSnet) context (Gramaglia 2015).

1.3. Prioritization and Validation Tasks: Rule-Based Solution

Hence, modernization impetuses move official statistics towards SOA in order to react promptly to ever-evolving scenarios and towards heterogeneous data integration to increase the level of quality in relation to some quality components and, possibly, the number of statistical outputs. Such impetuses should be taken into account when deciding IT solutions for a GSBPM activity, and specifically in relation to deterministic prioritization and validation tasks. In the latter case, besides GSBPM, relevant references are: (i) Generic Statistical Data Editing Models (GSDEM) (GSDEM 2015), which is a generic process framework for statistical data editing that focuses on the "Review and Validate" and "Edit and Impute" GSBPM phases; and (ii) specifically on the methodology for data validation (Di Zio et al. 2016), defined in the ESSnet context, which focuses on the "Review and Validate" GSBPM phase. In both cases, rule-based solutions are commonly used as methods for deterministic multiple source prioritization and validation tasks. In data validation "the decisional procedure is generally based on rules expressing the acceptable combinations of values" (Di Zio et al. 2016, 6), and in data editing "edit rules, score functions, correction rules and error localization rules" may be collected (GSDEM 2015, 15). Briefly, when rules are used, they are isolated from the software code, independently managed and customized by expert domain users, and may be accessed by different technological solutions, thus effectively executing the task. The rules are treated as data and not as parts of a source code of a program, thus enhancing rule re-use, sharing and increasing agility. Users with different roles may contribute to specific aspects

(e.g., the robustness of the validation rules in relation to a given task and quality objective, and also the robustness of the rules evaluation IT process).

1.4. Rule Engine

A rule-based infrastructure relies on a rule-processing engine (referred to as rule engine below), which is a component that evaluates which statistical unit meets the condition stated in a rule, and performs the corresponding imputation or correction actions, if any. In general, business solutions based on rules can be found in the literature in different contexts. Several examples can be found in the field of artificial intelligence, for the construction of expert systems (Lavrac 2001), in which expert knowledge in a given domain is represented with structures of the IF-THEN type (rules), which relate information or facts to some action. The architecture of an expert system includes a knowledge base, an inference engine, and a user interface that allows expert reviewers to interact with the facts and rules and maintain the system. Therefore, an expert system does not necessarily require the involvement of a database. The literature in this context define formal frameworks that study rule languages and define rule engines for processing specific language rules by assessing performance (Liang et al. 2009). Rule engines are also used in the literature to expand probabilistic knowledge bases (Chen and Wang 2014; Zhou et al. 2016). Currently, numerous processes of knowledge extraction from unstructured documents have also been proposed (De Sa et al. 2016), such as those available on the web, in order to build structured knowledge bases. Rule-based validation is also supported in the European Statistical System context, in which a formal framework for rule definition has been realized. An example is the Validation and Transformation Language (VTL), defined by an innovation project (Schafer 2015), which aims to be a single reference language for harmonizing the validation approaches among NSOs (Gramaglia 2015; Di Zio et al. 2016; ESSnet ValiDat Integration 2017). Currently, feasibility studies are being carried out in the ESSnet context to assess the possibility of defining converters from VTL to other languages used in national contexts, such as SQL (ESSnet ValiDat Integration 2017).

1.5. The Modern Solution for Rule-Based Prioritization and Validation Tasks

Current IT practices in deterministic rule-based validation tasks are outlined by Quensel-von Kalben (2017b). We adopt a SOA approach, by using a lightweight holistic micro-service (Fowler 2014). It is independently deployable and scalable. User interfaces allow expert staff to manage the rules and view the reports produced in relation to each rule-based task. The rule engine, which is the core of the service, is based on generalization and efficiency principles. It relies on an optimized data schema and on data parallelism in processing. It has sufficient efficiency to manage ever-evolving scenarios (i.e., small as well as big ones), and makes rule-based processing competitive with respect to other technological solutions. Therefore, a question arises whether rule-based tasks may exhibit the data selection agility property. Efficiency and agility in selection could be increased when inserting the input data selection logic within rules (i.e., selection rules) (Karpathiotakis et al. 2016). Therefore, selection rules may encapsulate the input source calls to services, which expose and allow access to input source data (i.e., referred to as

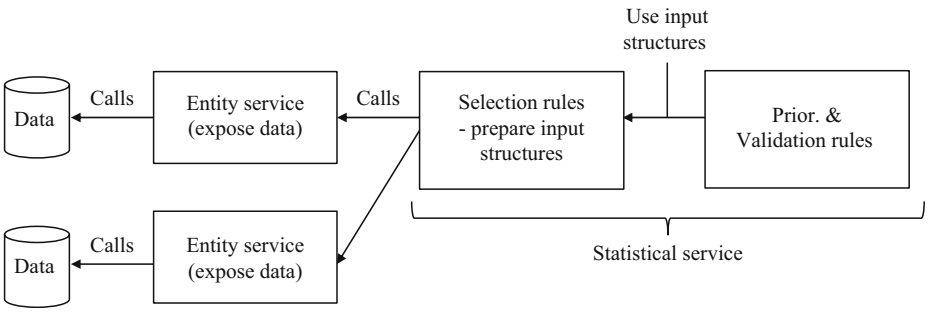

Fig. 1. Logical SOA architecture with Entity Services and Statistical Services: selection rules adapt input data for subsequent processing for prioritization and validation purposes.

Entity Services below) and adapt input data in structures for subsequent processing (i.e., the prioritization and validation rules evaluation). Entity Services may expose single as well as integrated datasets, and local as well as remote data, thus enabling architects to use data virtualization solutions in the input selection task. Entity Services expose the input data required by a Statistical Service, as sketched in Figure 1. Following virtualization principles could further increase agility. Selection rules could also interface with a logical integration layer or manage several source data clients in a transparent way for local and remote sourcing. Such rules allow the evaluation between data-replication architectures versus service-replication ones.

1.6. The Statistical Domain of Reference

The designed service has been used in a widespread manner in the Italian Statistical Business Registers (SBR) context. Briefly, Business Registers are updated yearly by integrating administrative and statistical sources, enabling identification of active statistical units and the estimation of the main structural, economical and identification variables for each unit using a robust methodology. Register data production may require the integration of an ever increasing number of evolving administrative sources and statistical lists, the integration of data from surveys, and integration of data from new unstructured web scale sources. In this evolving context, many deterministic prioritization and validation processes are needed, thus increasing quality standards (e.g., accuracy, comparability, coherence of a statistical output).

The rest of the article is organized as follows. In Section 2, we define the business and information models for deterministic integration and validation tasks. In Section 3 we briefly describe the use cases that concern the statistical user interactions with the IT system for rule and task report customization. We further describe the use cases for task processing, which involve the rule-processing engine. Specifically, we highlight the relevance of efficiency as a driver for providing usability and flexibility, and virtualization as a driver for increasing agility in input data sourcing. In Section 4, we show the benefits of an efficient and scalable rule-engine system and how the input data selection logic may be embedded within rules. We finally assess different technological solutions for remote sourcing. Conclusions are presented in Section 5.

2. Rule-Based Validation and Integration: The Business and Information Models

In this section we define the prioritization and validation service. In particular, we describe its abstract information model and give some hints on rule identification, particularly for selection purposes.

2.1. The Business and Information Models

Rule-based integration and validation tasks may be performed in several GSBPM phases. We refer to a generic modern process, as depicted in Figure 2. An IDentifier (ID) may be matched to any collected unit data in relation to a specific statistical population. By using the identification attributes, the unit may therefore be involved in the production process of a specific statistical output, integrating, possibly, multiple sources using common identifiers and requiring several processing steps. In rule-based integration and validation, specific-domain experts define the integration and validation rules, the necessary input variables from single or multiple sources, and the output variables useful for data validation or correction, which should be transmitted to a statistical output.

The tasks rely on a single base information set or *base table,* whose structure is depicted in Figure 3. Specifically, the *base table* is a statistical data set that is the object of the statistical process "Data prioritization and validation". "It is a collection of values. Conceptual metadata defines the meaning of these data by describing the concepts that are being measured by these data (concepts and definitions) and their practical implementation (value domains and data structure)" (GSDEM 2015, 13). Linkage and integration operations may be required to impute ID keys on source data before starting the integration and validation task. The *base table* fields include: (i) the statistical unit ID; (ii) other relevant secondary IDs, which may relate each unit with others for coupling or aggregation purposes; (iii) the input variables, which represent relevant unit characteristics; and (iv) possibly output variables (i.e., outcomes of the integration or

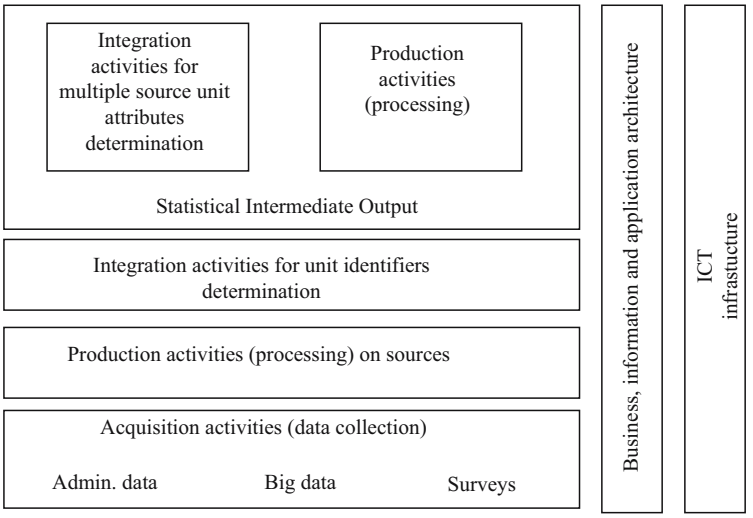

Fig. 2. Generic statistical process, which refers to GSBPM phases and GSIM terminology.

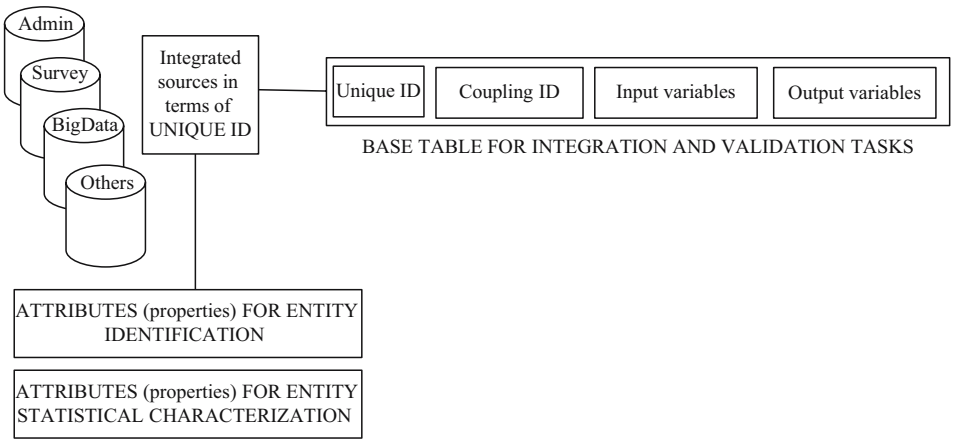

Fig. 3. Generic base table structure.

validation tasks, particularly in case imputation or editing actions are necessary). Rules are expressed in terms of the *base table* fields and any task relies on a single pair < rule set, base table >, that is, data structures and rules should rely on given metadata and be standardized. Metadata definition and common rule definition languages sustain sharing among domain experts. The *base table* is temporary and related to a specific task. Other downstream processes may transform the imputed, validated and/or edited values into a specific statistical output. Three different classes of rules may be specified:

1. *Selection rules*, which retrieve the necessary input data from sources (which may be local as well as remote with respect to the server that processes the validation task), using common ID keys, and define the *base table* input data for the validation task,
2. *Indicator rules*, which determine whether a given condition is met for each unit,
3. *Imputation rules,* which impute a specific value to a given variable under specific conditions.

Generally, GSDEM edit rules (which describe valid or plausible values for *base table* variables or *base table* combination of variables, and detect values presumed to be in error). GSDEM score functions evaluate input data values at unit level and GSDEM error localization rules presumed to be in error without a detectable cause. All these elements may be expressed through indicator rules. Moreover, GSDEM correction rules amend errors and may be expressed through imputation rules. Therefore, rules may exhibit GSDEM review, selection and amendment functions (GSDEM 2015, 8). Moreover, as outlined in Di Zio et al. (2016, 10), several validation levels may exist that review the logical and statistical consistency of the data and that involve more and more input information. Selection rules enable raising the validation level in relation to the business. Rules are expressed in a declarative way. Indicator and imputation rules rely on SQL clauses. Selection rules, when remote sourcing is used, may be based on either SQL and XML. We call them Xrules. The rules, formally specified by the statistical domain experts, are evaluated (i.e., applied) when sources are available. Each rule is uniquely identified (i.e., Unique ID, and Process ID in Figure 4), may or may not exhibit a selective condition

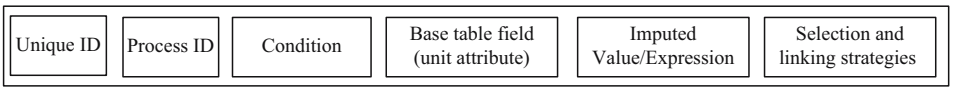

Fig. 4. Generic rule structure.

of validity in relation to the units within the *base table* (i.e., condition in Figure 4), may or may not set a given *base table* field, whose value may be the result of a specific combination of the input variables, or the result of aggregation operations computed on other *base table* fields (i.e., *base table* field unit attribute and imputed value in Figure 4). Xrules may be further based on a triple: (i) the call to a remote Entity Service, whose client has to be available to the calling server, for retrieving source data; (ii) a selection clause for taking into account only relevant information in virtually loaded remote data sets; and (iii) a linking clause for matching virtually loaded units with those within the *base table*. The rule-based integration and validation service may correspond to 5.1 (Integrate), 5.3 (Review and validate), 5.4 (Edit and Impute) and 5.5 (Derive new variables and statistical units) phases of the GSBPM (ESSnet 2015). A few relevant extracts from the GSBPM documentation (ESSnet 2015, 18) are given to assist the reader as follows:

"The 5.1 subprocess integrates data from one or more sources. The input data can be from a mixture of external or internal data sources, and a variety of collection modes, including extracts of administrative data. The result is a harmonized data set. Data integration typically includes:

1. matching/record linkage routines, with the aim of linking data from different sources, where those data refer to the same unit,
2. prioritizing, when two or more sources contain data for the same variable (with potentially different values)."

The integration phases may be performed sequentially. Specifically, we focus on the deterministic prioritization of data from different sources and on linking operations, which rely on unique units' IDs. Probabilistic or more complex record linkage operations are outside the scope of this document. They have to be performed elsewhere in the process chain.

The abstract information objects required by a generic integration service are shown in Figure 5. Integration evaluates conflicting source data and sets a single chosen value in output variables. The task returns the imputed data and a monitoring report.

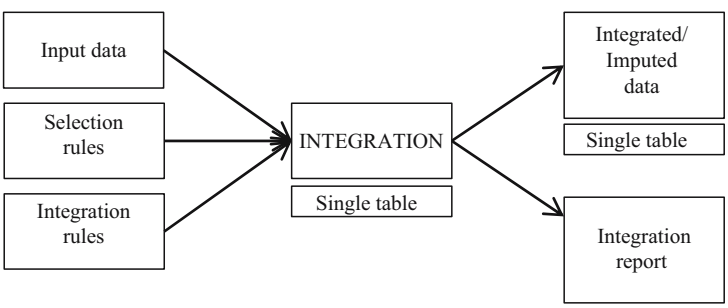

Fig. 5. Abstract information model of a general integration process.

Phase 5.3 "Review and Validate" of the GSBPM document (ESSnet 2015, 18) states;

"This subprocess applies to collected micro-data, and looks at each record to try to identify (and where necessary correct) potential problems, errors and discrepancies such as outliers, item non-response and miscoding. It can also be referred to as input data validation. It may be run iteratively, validating data against predefined edit rules, usually in a set order. It may apply automatic edits, or raise alerts for manual inspection and correction of the data. Reviewing, validating and editing can apply to unit records both from surveys and administrative sources, before and after integration. In certain cases, imputation (phase 5.4) may be used as a form of editing".

With respect to phase 5.4 "Edit and Impute", (ESSnet 2015, 19) it states:

"Where data are missing or unreliable, estimates may be imputed, often using a rule-based approach. Specific steps typically include: (i) the identification of potential errors and gaps; (ii) the selection of data to include or exclude from imputation routines; (iii) imputation using one or more predefined methods for example "hot-deck" or "cold-deck"; (iv) writing the imputed data back to the data set, and flagging them as imputed; and (v) the production of metadata on the imputation process";

And finally, phase 5.5, "Derive new variables and units", (ESSnet 2015, 19) is described as follows:

"This subprocess derives (values for) variables and statistical units that are not explicitly provided in the collection, but are needed to deliver the required outputs. It derives new variables by applying arithmetic formulae to one or more of the variables that are already present in the dataset. This may need to be iterative, as some derived variables may themselves be based on other derived variables. It is therefore important to ensure that variables are derived in the correct order. New statistical units may be derived by aggregating or splitting data for collection units, or by various other estimation methods. Examples include deriving households where the collection units are persons, or enterprises where the collection units are legal units".

Validation rules encapsulate deterministic conditions and actions, and may be expressed as algebraic expressions. In Figure 6, we sketch the main information objects that support the rule-based validation process. Validated data, a validation report, and further automatically edited data, whenever editing is possible through deterministic rules, are the main outcomes of the task. When manual editing is necessary, the output reports assist the expert user.

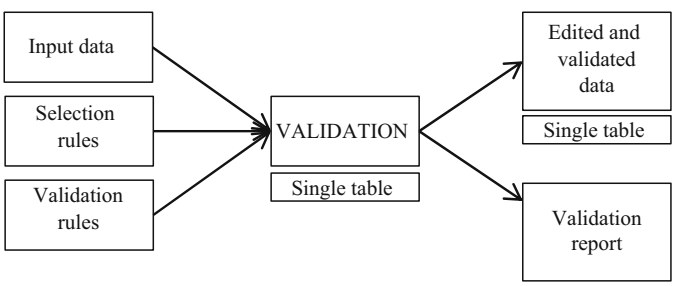

Fig. 6. Abstract information model of a general validation process.

2.2. *Hints for Rule Identification*

The logical flow of these processes has already been described. All units of the reference statistical population are imported into a single *base table*. Selection rules, through *existence* operators, enable the user to relate the *base table* with local or remote sources, storing temporarily input values to be used for validation and/or transformation tasks. Expert statisticians express their knowledge in terms of task rules, which are then applied on the input data. It is out of scope of the present document to show classical prioritization or validation rules, which evaluate different fields of the *base table* for detecting wrong or misleading values: data structures and rules depend on the specific domain and should be shared, based on metadata and possibly standardized. For details, the reader may refer to Di Zio et al. (2016) and GSDEM (2015) for an indepth analysis. We remark that, in practice, a large part of the deterministic rule requirements are satisfied through the use of logical, existence and inclusion operators. Moreover, the same rules could also be described in VTL for sharing and documentation purposes.

A sample selection rule is shown in Figure 7a. More complex queries may be required, for example when aggregation or algebraic functions must be performed on disjoint groups of units. A sketch of a generic aggregation rule on disjoint groups of units, identified by a given key, SETID, is shown in Figure 7b. These types of rules may also be used to redistribute some variables under certain conditions and to compute derived variables, for proportional inference, source prioritization, and integration and correction purposes, as explicitly outlined in Subsection 2.1. An example of a distribution rule may be as follows: a first derived variable sums up other variable values in relation to disjoint groups of the statistical units.

A second variable computes the number of units actively involved in any group, and finally a third variable represents the proportional imputation of the summed-up variable, that is, the proportional distribution of the aggregated variable in equal parts on the units actively involved in the group.

3. Micro-Service Architecture, Agile Cooperation, Efficiency and Data Virtualization for Service Reuse and Sharing

In this section, we highlight generalization, service efficiency and agility as application design principles sustaining flexibility, and hence service reuse and sharing. In particular, the micro-service architectural pattern, the agile cooperation, and data virtualization solutions may sustain service agility. An extensive performance assessment may evaluate service efficiency. The promoted design pattern, which relies on service autonomy in evolution and deployment and on efficiency in processing, has enabled widespread usage

Sample selection rule (a)	Sample aggregation rule (b)
Update BaseTable base Set field1 = (select genericField from SourceTable source where base.ID = source.ID) Where exists(select 1 fromSourceTable source where base.ID = source.ID)	Update BaseTable base Set field1 = (select sum(genericField) from BaseTable source where base.SETID = source.SETID) Where exists(select 1 fromBaseTable source where base.SETID = source.SETID)

Fig. 7. (a) Sample selection rule. (b) Sample aggregate rule on disjoint set of rules.

of the prioritization and validation service in the SBR domain. The highlighted principles are furthermore CSPA-compliant, and potentially facilitate service-sharing even across domains and organizations. CSPA principles sustain an inclusive design of components. Inclusion with respect to various domains, ever-evolving scenarios and input/output communications. When service flexibility is increased, a service might potentially be used in a widespread manner.

3.1. Micro-Service Architecture and Agile Cooperation

The implemented service relies on the micro-service architectural pattern: it is a lightweight basic one and relies on its own web user interfaces, a self-contained schema and an efficient rule-processing engine. All functionalities and data of a specific business capability are realized by an independent service, which can be deployed on specific hosts (Fowler 2014). A natural distribution of the workload sustains system efficiency and availability. In the case of increasing load, micro-service relocation and/or replication on a cluster, or in the cloud, may assure scalability. The deterministic integration and validation tasks have been vertically solved in an autonomous and stateless manner. The service takes the input data, processes the rules and generates the output reports. When it fails, it can be restarted without any dependence on previous states. The service is unaware of its position in the process chain/control flow. The service, which provides holistic system functionality, is independently deployable both with respect to user interfaces and to processing components, and may be independently progressed in both cases. In this section, we specifically describe the interfaces for user interaction (i.e., the use cases that involve web user interfaces). We describe data management and processing issues (i.e., the use cases that involve different actors with regard to the service users and relate specifically with task execution) in the following sections. Micro-service pattern sustains short agile software development cycles in a stepwise manner, and by involving users, also in the functional design phase and in the acceptance testing phases. Such development pattern could decrease difficulties in service reuse. In particular, through the developed web application, the domain expert users may manage and customize the task rules, view the output reports (i.e., the outcomes of the executed tasks), as well as download specific information sets in relation to the statistical units, whose characteristics met the condition of a rule during processing, and possibly manage metadata reporting. The user interfaces rely on a specialized Java application, based on a software architectural pattern similar to that used in the context of the COmmon Reference Environment (CORE) Project – Eurostat (Scannapieco et al. 2011). An expert user may insert new rules and modify or delete existing ones through specific web functionalities. Any rule is equipped with a customizable text field used for documentation purposes: it might also contain the equivalent VTL description, thus using a *lingua franca* to describe it. As already outlined, rule-based integration and validation provides an effective decoupling between the domain expert work and the IT work, providing flexibility in rules definition. Web functionalities for executing and monitoring rule-based processing could further increase the ability of users to run the service with less reliance on IT experts. A generic output report shows the number of units that met the rule conditions during rule-processing, and, if necessary, it may decompose such total number into subsets by classifying the units in

Report Title – ID PROCESS							
Id Rule	Level_rule	Condition_rule	Description_rule	Count_all	Subcount_col1	Subcount_col2	…
1	1	Example: SQL where clause	It could be the VTL description of the rule or a natural language one	501	101	203	197
……	……	……	……	……	……	……	……
Downloadable Lists Section – ID PROCESS & ID REPORT							

«Application Frequency» Values For Each Rule

Fig. 8. A skeleton sample output report (final check in SBR context): the overall number of units for which the rule condition triggered is shown. Such value is decomposed in relation to four disjoint classes of the involved units.

relation to relevant classes of data (e.g., in SBR validation context, the report may classify the outcomes in relation to not-active legal units and active legal units in relation to specific ranges of employees). Different reports, which sub-classify data differently, may exist for the same homogeneous set of rules. The report may be consulted as well as downloaded by expert users for further analysis, thus enabling the users to check which statistical units contributed to the count. Each unit in the downloaded lists may be equipped with a customizable information set (i.e., other relevant variables at unit level). Figure 8 shows a typical output report. Each row corresponds to a single rule. The total rule-validity frequency with respect to the involved statistical units is shown. Furthermore, such frequency value is decomposed in relation to disjoint classes of units.

3.2. Data Schemas Solutions for Highly-Performant Data Management

A data schema for supporting general rule-based processes has been designed. Generalization is preserved by parameterizing the concepts of interest, namely (i) the specific process/task associated with a fixed *base table* structure and rule set; (ii) the statistical units typology; (iii) the specific output report; and (iv) the downloadable custom information set. Generally, integration and validation tasks may involve millions of units, or more, and may also be based on hundreds of control/processing rules. In this scenario, each server, which hosts the rule-processing engine, may manage simultaneously numerous integration and validation tasks. Therefore, a highly-performant data management, which relies on efficiency principles, should be used. In classical theory, BI applications need to evaluate the tradeoff between minimization of data duplication and the increase in data duplication for reducing the cost of expensive integration operations among separated data structures (Pullokkaran 2013). In rule-based tasks, a question arises whether a schema, where duplication is carried on, may improve performance in processing and downloading by allowing each task to rely on a single data structure that maintains all the necessary information for a single task.

Specifically, the association between a valid rule and a statistical unit, whose characteristics met the rule condition during processing, may be stored in a separate data structure (i.e., a join table), as sketched in Figure 9, the same for each task, or in the *base*

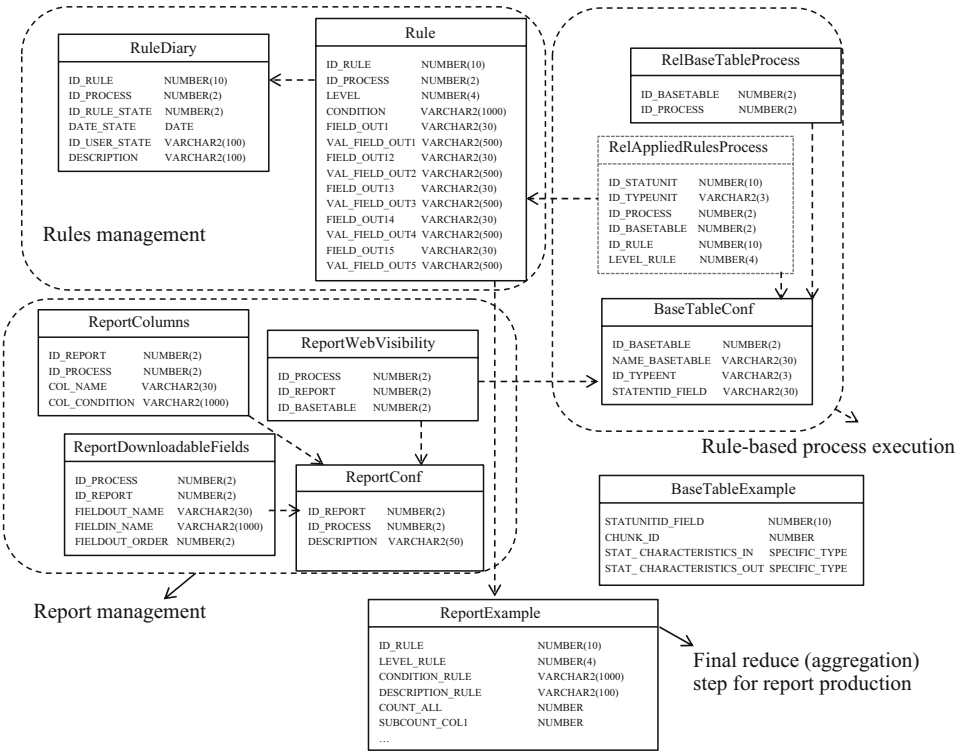

Fig. 9. Skeleton ER for the schema with less duplication.

table itself, which is specific and different for each task. The common join table may soon accommodate tens of billions of records. It should be organized to provide data proximity: specifically processing and downloading operations should access only opportunely partitioned and close data. The improvement in the selection queries time behavior positively affects overall download performance, and in addition, enhances the user experience.

3.3. The Parallel Rule-Engine

The designed integration/validation processes are *de facto* heavily parallelizable: they work on a specific population of statistical units, retrieve all the needed information and execute the set of rules of interest in relation to each unit or to disjoint grouped unit sets. Therefore, data may be divided into many fine grained similar tasks (i.e., which realize the same operations, i.e., the same rules, on disjoint sets of data) and output reports in relation to each subset of data may be aggregated into a single final report. Parallelism may refer to task parallelism as described by Subhlok et al. (1993) and data parallelism, which focuses on distributing the data across different parallel computing nodes. We explore the latter technique for developing an efficient rule engine solution. Parallelism is generally used in clustered systems, where performance preservation is granted by saving server resources and by giving equal conditions to all tasks in which a job is massively executed in parallel (Ananthanarayanan 2013; Ananthanarayanan et al. 2013). In order to solve such problems,

recent studies propose several techniques to determine the impact of parallelism on the amount of resources (Delimitrou and Kozyrakis 2014). Platform dependent algorithms for managing the parallel execution are called cache-aware algorithms (Prokop 1999). They are particularly relevant in database management server heavy load conditions, when the benefits of parallelism start to fade. The proposed rule-processing engine solution for integration and validation tasks in official statistics domains makes use of data parallelism as follows. Data are divided into consistent subsets (i.e., *base table* bands or chunks associated to similar mini (i.e., sub-) tasks); a flow of similar mini tasks is hence provided to a given number of active server processes. The rule-processing strategy is depicted in Figure 10. Through such a mechanism, only a subset of the whole set of data is managed by the database management server in a given time unit and, if the mini tasks are solved in an efficient manner, the cache stress is constrained.

Specifically, the number of parallel servers and the chunk dimension may be set optimally, thus saving database management server resource consumption. We will show system benefits in terms of resource consumption in the following section.

3.4. Selection Logic Embedded in Rules

In this section, we focus on the input data selection sub-task. When designing a service for solving multiple sources prioritization and validation tasks, an important issue is how to retrieve input data for performing these tasks. Our Statistical Service may be hosted in specialized servers, possibly scalable ones. However, input data may be spread in several intra/inter NSOs remote systems, which could be based on different technologies. Dedicated integration solutions, which load *ad-hoc* the necessary inputs in the same homogeneous enviroment, may bring to data silos and data labyrinth (Van Der Lans 2013). Moreover, data locality may require the definition of expensive processes (Goede 2011) and a careful design of data replication-based architectures. As outlined in Section 1, in order to avoid such silos, the literature proposes data virtualization for ever-changing integration needs (Krawatzeck et al. 2015). Remote source-independent selection becomes a key element for ensuring agile data integration. In the European context such solutions

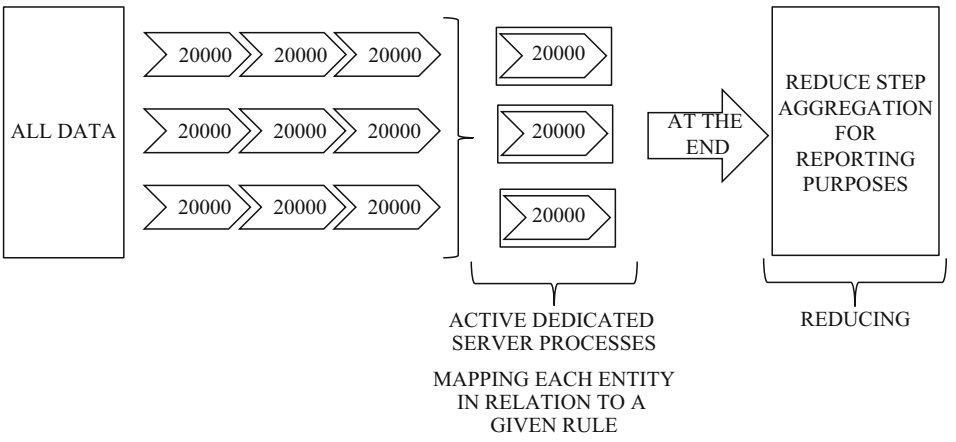

Fig. 10. Data parallelism strategy.

have also been increasingly explored for modernizing inter/intra NSO production (Gramaglia 2015). In these cases, security and confidentiality issues (Yu et al. 2010; Zissis and Lekkas 2012), in relation to "Not only Auth-entication and Authorization" (NoAuth) issues, are outside the scope of this document. However, particularly when Entity and Statistical services are exposed outside a single trusted domain (e.g., in inter-NSOs scenarios or in case web-linked external resources are used in the data production chain), they should be taken into account and analysed in-depth. Specifically, we have explored the usage of the implemented rule engine for the selection issues. Data parallelism may increase efficiency in selection, thus opening up various data architectures, where data may be stored locally with respect to the server that hosts the rule engine, as well as remotely. In fact, remote sourcing may involve multiple servers. A large amount of exchanged data stresses the multiple servers resources and may be a stumbling block in using such a solution. Remote sourcing may benefit from a more efficient data exchange. We encapsulate source data calls within the rules. Each different data source consists of a different remote call. In such a scenario the server, which hosts the rule-processing engine, must use the data source callable functions (i.e., clients). The available clients represent our remote integration set, which may be virtually integrated in a temporary *base table*. Conversely, selection rules might interface with a separate single integration layer. Different technologies enable architects to promote remote sourcing. We consider two different solutions, thus outlining robust considerations in relation to data virtualization use in the statistical context. Specifically, distributed connectivity may be provided by using database connectivity technologies in homogeneous and heterogeneous environments, possibly by relying on gateway agents and drivers. Otherwise, distributed connectivity may be provided by using web services, thus providing the highest level of interoperability in heterogeneous environments. Each different component framework, using wrapping, may be exposed through web services. In particular, we assess the following technologies: database links in an Oracle homogeneous environment, and Simple Object Access Protocol (SOAP) web services, exchanging data using eXtended Markup Language (XML) format, which is a standard for data and message exchange over the internet.

4. Highly-Performant Data Management, Efficiency in Processing and Virtualization in Selection Assessment

In this section, we assess the robustness and efficiency of the designed service by tackling the open issues arising from the previous sections. In particular, our experimental results show how engineered highly-performant data management together with fine-tuned parallelism techniques may substantially improve the flexible, inclusive usage and therefore the level of reuse of a prioritization and validation service. We also compare the aforementioned input data selection solutions by assessing local data with respect to remote data access. In the latter case, we assess the usage of database connectivity technology with respect to web service technology. Data locality is obviously the best choice in terms of processing times. However, data locality may require static loading processes. The cost of the data replication architecture should also be taken into account. Parallelism can mitigate remote-sourcing worse performance; likewise, service replication architectures, which rely on scalable services, should be assessed.

4.1. How Relevant is a Performant Data Management for Having Sharable Services?

In this section we evaluate different data schema solutions. We validate the highly-performant one in relation to the service operations. We show the benefits of data duplication reduction and of optimized data schemas in task processing. In particular, in Figure 11 we show the unitary execution times in milliseconds (i.e., the ratio between the execution time of a process and the number of statistical units involved) of a single rule-based process by using the less-duplication (i.e., partitioned with less duplication) schema and the more-duplication one, as outlined in Subsection 3.2, when the number of involved units increases. The association table between applied rules and statistical units accommodates the data of a single process. The tested process is an SBR validation process based on the evaluation of about 400 rules. The results clearly show that the data schema with less duplication (double dashed line) outperforms the other one (solid line).

Figure 11 also shows that the unitary execution time of a rule-based process remains almost constant as the number of involved units increases (in normal database load conditions). The execution times curve is linear in the number of statistical units (obviously, when the load conditions increase and physical server resources become scarce, the performance decreases and the curve changes its shape). We measured similar processing times in relation to the less-duplication schema, even in the case of eight simultaneous validation processes, operating on different *base table*s, but on similar data and rules, and resulting in about 2,000,000,000 records in the common join table. A highly-performant data management increases the flexibility in service use. In the next section, we assess a further improvement in performance by enabling the system to use scalable processing techniques.

4.2. The Parallel Engine

In this section we show the benefits of using parallelism. Opportune settings of parallel execution parameters may produce efficiency gains in terms of both execution time and

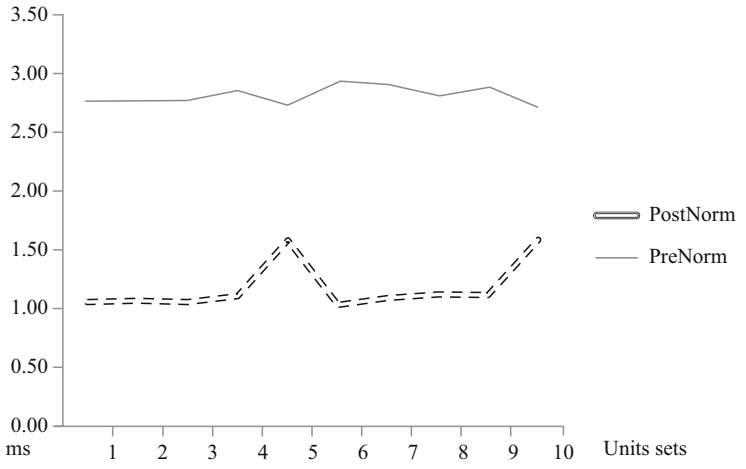

Fig. 11. Unitary execution times (in ms) (Y-axis) of ten executions (X-axis) of a rule-based process when a data schema with more duplication (solid line) and a schema with less duplication (double dashed line) are used. On X-axis the number of involved units increases from about 1,370,000 up to 13,700,000 units (ten executions).

server resource consumption. We assess typical performance measures of an Oracle database management server: the task execution time, the consistent gets, the Process Global memory Area consumption (PGA) and CPU time consumption.

Briefly, consistent gets represent the number of logical read requests to get data from the memory area shared by all the processes. PGA represents the single process dedicated memory area size and it is a dynamic, limited part of the overall shared one. In Figures 12 and 13 we show the time behavior (i.e., execution time curve) of a big validation process (around 13,000,000 statistical units and 400 validation rules) in relation to an increasing number of simultaneous active server processes when parallelism is used. In Figure 12, the dashed line shows the execution time of a chunked validation task when a single server is allocated, divided by the X-axis number of servers. The latter refers to an ideal parallel process, since it represents the ideal situation when several parallel processes solve the same task on disjoint subsets of data. The dotted line refers to the actual experienced execution time of the chunked validation task when allocating from one to five active parallel servers. The solid line shows the execution time of an equivalent non-parallel process, which may be computed by multiplying the actual execution time of the parallel validation task (i.e., the dotted line values) by the number of allocated parallel servers (i.e., X-axis numbers). Parallelism is effective in reducing the processing time of the task and close to the ideal case in relation to the single server time. Furthermore, in Figure 13a we show the scalability level of the parallel executions and their robustness in relation to several load scenarios.

In particular, in Figure 13a and 13b *Parallel_P1*, *Parallel_P2*, *Parallel_P3* represent three simultaneous "big" parallel similar validation processes, which solve the validation task in relation to the same input data, units and rules. The word "big" refers to a validation process with the same set of rules (approximately 400) and related to the same statistical population, about 13,000,000 statistical units, although working on different *base tables* and metadata. *Parallel_NoP* is a single parallel execution, processed in a separate test session when two other big, non-parallel similar (as before) validation processes are active

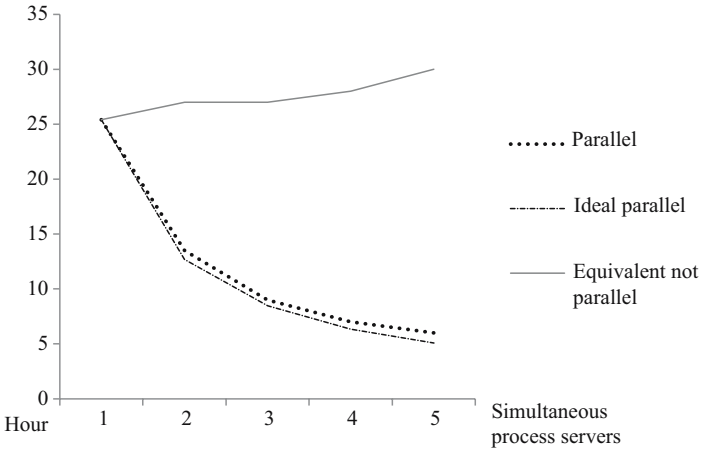

Fig. 12. Real rule-based process executed in a parallel fashion in relation to an increasing number of active allocated server processes with respect to an ideal, thus optimal, execution of a rule-based process in parallel fashion.

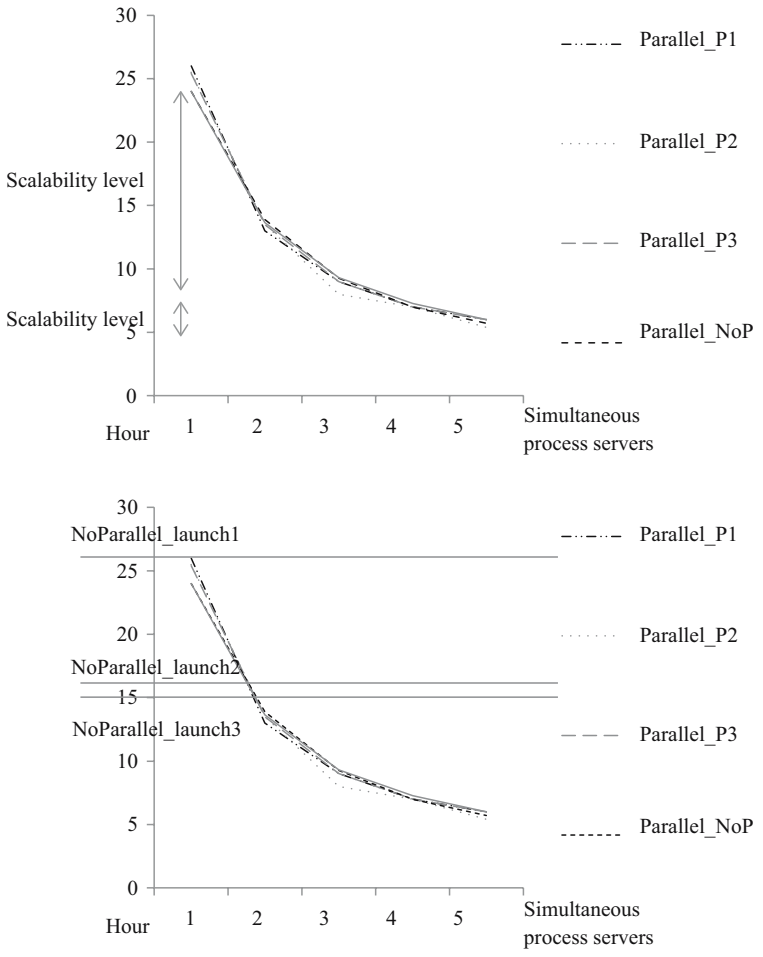

Fig. 13. (a) upper, scalability level and speed up in relation to the active allocated server processes; (b) lower, several parallel execution of the same (same data, same rules) rule-based process by increasing the number of allocated parallel servers. The straight lines aim to point the execution times of three contemporary executions of the same rule-based process, processed in a non-parallel fashion (obviously not varying in relation to the number of allocated servers).

at the same time. *NoParallel_launch1*, *NoParallel_launch2*, *NoParallel_launch3* are three simultaneous executions of three big similar validation tasks, when parallelism and chunking of data are not used. The scalability level substantially decreases when the numbers of allocated simultaneous servers grows. The execution time gain is therefore smaller and smaller in relation to an increasing number of active parallel servers. The minimum number of active servers in relation to a target speedup and to the server resources preservation is a rewarding processing choice. Each active server process manages a mini task in a given time unit: the fewer active mini task/processes we allocate, the lesser resources we simultaneously consume. Moreover, parallel executions performance does not vary in relation to different load scenarios. The performance of the parallel processes is quite stable, even when other "big" processes (parallel or non-parallel) are executed. On the other hand, one of the three simultaneous non-parallel big

Table 1. Local data: mean performance indices in relation to six non-parallel executions of a selection process on 28 different sources (i.e., the process is composed by 28 different selection rules) and related to an increasing set of involved statistical units (from 10,000 to 60,000) and in relation to the corresponding parallel executions (with four and six servers).

	LOCAL DATA			
	Mean single chunk (A)	Ideal single chunk	Mean not-parallel execution (B)	B/A
Consistent gets	509058	346726	3120532	6.13
CPU	631	4883	15675	24.84
PGA	2696481	494110	4446995	1.65

processes experienced decrease in performance. In Figure 13b, when just one active server is allocated for parallel execution, by letting it manage the overall flow of chunked validation mini-tasks, performance is worse than the single non-parallel execution. The parallel processing performance is driven by the single unit overhead, introduced by chunking. Each single chunk of data introduces an overhead in processing due to a not ideal consumption of resources. Therefore, in Table 1 we show the server resource consumption in terms of consistent gets, CPU time consumption in centiseconds and PGA consumption in bytes in relation to a single processing task in case of non-parallel execution and to a single processing mini task in case parallelism is used.

Specifically, we evaluate performance indices for a selection task (i.e., processing of the selection rules for retrieving input data for a specific integration and validation task), which is related to 10,000, 20,000, 30,000, 40,000, 50,000 and 60,000 statistical units, and which has been processed in a parallel (both four and six contemporary active servers and mean chunk dimension is about 4,000 units) and non-parallel fashion. In Table 1, the mean values are obtained by averaging both the non-parallel and the parallel executions. B/A outlines, in a given time unit, the over-consumption in relation to a specific index of performance of a mean single/unique non-parallel execution with respect to a mean single parallel mini task execution. It shows a possible degree of parallelism (i.e., number of simultaneous active mini tasks) we may choose for consuming in a given time unit less resources in relation to the single/unique non-parallel execution of the same overall process. Consistent gets and PGA refer to the server memory cache. PGA seems to be the more critical parameter. Benefits in time execution might require an over-consumption of single process server memory. The scalability issue becomes a relevant topic.

In Figure 14, we show the PGA used and consistent gets for one single statistical unit when processing selection rules in parallel (black line in the figure) and in non-parallel (grey line in the figure) fashion in relation to an increasing number of involved units, as before. The overhead of the parallel process in unitary terms determines the performance decrease between the non-parallel single/unique execution and the chunked parallel exeuction when only one active server is allocated for managing the overall flow of chunked mini tasks, as shown in Figure 13b. Therefore, optimization in data schema design, described in Subsection 3.2, is even more relevant in relation to the unitary over-consumption of server resources, thus increasing the benefits of data parallelism. The consistent gets overhead appears linear in the selection dimension, while PGA

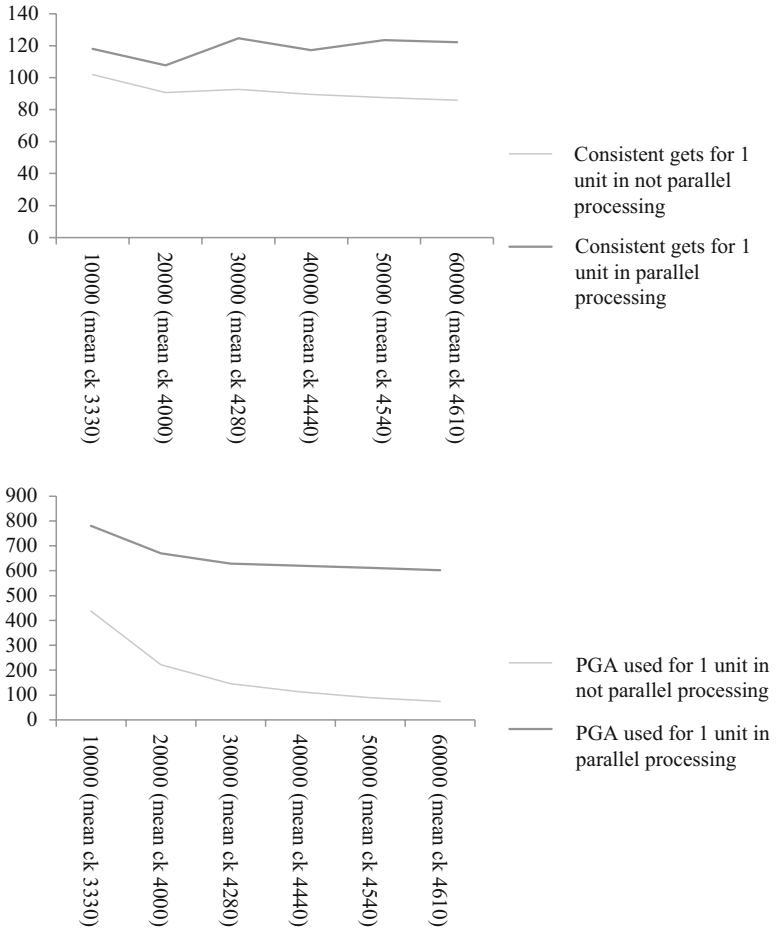

Fig. 14. Unitary PGA used in bytes and unitary consistent gets in number of data block reads for a selection process which is performed in parallel and in non-parallel fashion.

consumption does not seem to exhibit the same trend. Linear trends produce a fixed single server performance decrease, which guides the performance of the overall parallel processing, whatever chunk dimension we choice.

4.3. Data Loading for Prioritization and Validation Tasks: Where Can We Place Input Data?

Nowadays, data-driven architectures have been increasingly imposed to migrate old IT systems or build new ones in a SOA perspective. We explore the use of the developed efficient rule engine for input selection purposes by encapsulating within selection rules the input data calls. In this section, therefore, we assess the data architectures presented in Subsection 3.4. Specifically, we compare local data (with respect to the server that hosts the rule-processing engine component) and remote data sourcing in terms of selection time and server resource consumption by using the same performance measures presented in the previous section. In the case of remote sourcing, we focus on a distributed database

scenario, by comparing database connectivity technology for exchanging data (i.e., referred to as JDBC/SQL below), and a web-serviced data scenario (i.e., referred to as SOAP/XML below), by using web services for exchanging data in XML format. The latter solution enables architects to use interoperable Entity Services for sourcing any Statistical Services. In Figure 15, we compare selection times in six different scenarios and in relation to an increasing number of statistical units. On the X-axis we have different sets of involved units from 10,000 up to 60,000. In Figure 15, the *dblink* curve refers to a non-parallel execution of a selection task, which is composed of 28 selection rules, when a JDBC/SQL selection is adopted. The *dblink p4* one refers to the execution of the selection process in a parallel fashion, by using four active simultaneous process servers, when database connectivity technology is used.

The *dblink p6* curve refers to a processing scenario with six simultaneous active servers. The curve labelled with *xml* represents the non-parallel processing of the same selection tasks when data are exchanged through XML. The *xml p4* curve refers to the parallel processing of the selection tasks with four active servers and finally the *xml p6* curve refers to the parallel processing of the selection tasks with six active simultaneous servers. The curve that is labelled with *local* refers to the same tasks as before when source data are stored locally with respect to the server hosting the rule-processing engine and parallelism and data chunking are not used. Local data obviously exhibits the best performance, but the static loading issue remains. XML data exchange time is affected by the XML serialization and deserialization processes and by the selection of virtually loaded data, before importing them into the *base table*. Therefore, it exhibits the worst performance in relation to the rule-based selection tasks, although parallel execution may partially mitigate the performance decrease. The knowledge of the above curve may help architects to choose the most suitable selection scenario, possibly adopting the most interoperable data exchange both in intra-NSO and in inter-NSOs context, when the overhead is acceptable or may be suitably managed. We assess server resources consumption in

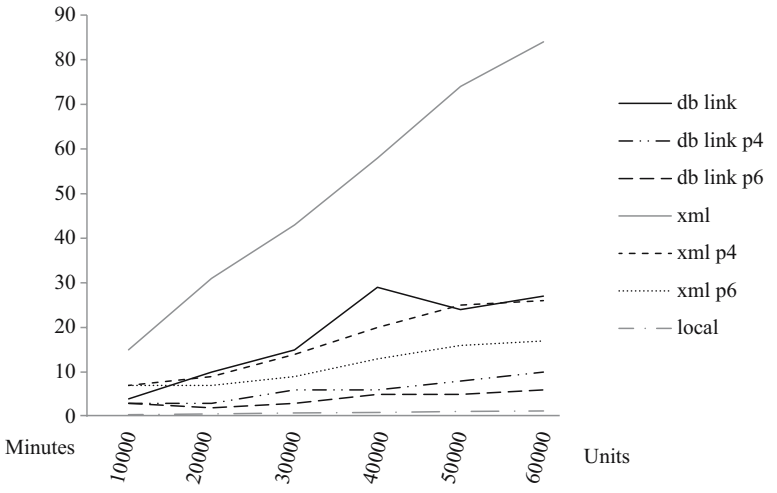

Fig. 15. Selection times in case of database selection (three scenarios: non-parallel, parallel with four servers and parallel with six servers) and in case of xml selection (three scenarios: non-parallel, parallel with four servers and parallel with six servers).

Table 2. Specifically, CPU time, PGA consumption and shared cache accesses (i.e., consistent gets) are presented.

The benefits of XML data exchanges are counterbalanced by higher resource consumption in terms of cache stress and CPU time. The SOAP/XML overhead in resource consumption is always greater than the JDBC/SQL one. Agile database deployment and service scalability, which is further recommended in a micro-service architecture, might surely be relevant drivers for enabling architects to choose data virtualization solutions based on XML data exchange. Entity Services and Statistical Services should be easily scalable. Container-based architectures (Xavier et al. 2013), which moreover ensure the technology neutrality property in service development, seem promising in providing agility in database deployment. They sustain scalable service architectures and should be explored, thus supporting a future-proof manner Statistical Service sharing. When remote sourcing is used, the overhead in resource consumption, for example, in terms of memory, is higher than when data locality is granted. However, in SOAP/XML scenario the degree of parallelism we may set, consuming less resources with respect to the corresponding execution when data parallelism and data chunking are not used, is higher than in JDBC/SQL one. Specifically, due to data parallelism, the speed-up gain may be substantial in the case of XML data exchange, as Figure 15 shows, thus enabling architects to choose such selection scenario in a managed way. Data virtualization benefits may be achieved when a performant data exchange is ensured. Data parallelism and selection rules sustain selection performance. Scalable service architectures may manage any over-consumption of involved servers resources.

Table 2. Mean performance index values with regard to the parallel and non-parallel executions, whose processing times have been shown in Figure 14.

JDBC/SQL DATA			
	Mean single chunk (A)	Mean not-parallel execution (B)	B/A
Consistent gets	510323	287752	0.56
CPU	2920	46259	15.84
PGA	2961426	5222504	1.76
SOAP/XML DATA			
	Mean single chunk (A)	Mean not-parallel execution (B)	B/A
Consistent gets	904636	5158917	5.70
CPU	24067	699154	20.52
PGA	18359343	27679507	1.51
OVERHEAD RESOURCES BETWEEN SOAP/XML AND JDBC/SQL %			
	Mean single chunk (A)	Mean not-parallel execution (B)	
Consistent gets	43	94	-
CPU	91	93	-
PGA	83	81	-

5. Conclusions and Future Work

5.1. Summary of the Proposed Solution

In this article we describe a multiple source rule-based prioritization and validation service, successfully developed in the Italian SBR context. Rule-based solutions provide decoupling between the domain experts and the IT experts, sustain agile rules evolution and simplify sharing. We assess a micro-service solution, which may be easily inserted into a production chain and is an affordable migration path towards a SOA. It also facilitates the agile cooperation between domain expert users and IT ones. We specifically promote optimized data management and efficient data processing, using data parallelism techniques, in deterministic rule-based tasks. We further assess selection rules, which encapsulate the input-source calls to Entity Services. The latter might expose single as well as integrated datasets, local as well as remote data, thus enabling architects to use data virtualization solutions in the input selection task. Data locality obviously shows best performance, although the static data loading problem remains, and data replication architectures and consistency issues should be carefully taken into account. Remote sourcing, when needed, requires attention in physical server dimensioning and scalability issues.

5.2. Highlighted Key Principles

Modernization impetuses move official statistics towards reuse and sharing of methods and components. Specifically, it moves official statistics towards SOA, in order to react promptly to ever-evolving scenarios and towards heterogeneous data integration, in order to increase the level of quality in relation to some quality components and, possibly, the number of the statistical outputs. Such impetuses should be taken into account when deciding IT solutions for a GSBPM activity. We aim to promote an inclusive application design pattern which enables reuse and sharing in a modern way. Specifically, we promote the following, CSPA compliant, principles. The "single capability principle", which is a functional foundation of micro-service architecture, ensures the minimization of costs for new or changed requirements. The "technological neutrality principle", which does not impose a specific development, integration or deployment platform. The micro-service architecture does not rely on a given technological platform, but rather on stateless, autonomous and self-contained data schema, web-user interfaces, and processing components, which may be independently deployed. We further highlight the importance of non-functional requirements. In particular, performance assessment and efficiency evaluation are relevant drivers for an inclusive reuse of the service. Data virtualization is another relevant driver which increases agility. It enables a SOA where Statistical Services may call remote Entity Services to consume input, eventually integrated, data.

5.3. Future Work

Care should be taken in implementing the micro-service architectural pattern: it introduces some extra administrative overhead, in particular for deployment, administration, monitoring and security. When data virtualization is used in a single trusted domain,

Auth-entication and Auth-orization (Auth) policies and techniques are well defined and easily taken into account in service usage. However, when interfacing various inter-organization domains, federated Auth policies should be engineered and NoAuth issues, such as the confidentiality one, should be taken into account as strict nonfunctional requirements which simplify service sharing. Security issues could be stumbling blocks in sharing. Even technological issues may represent a stumbling block in service sharing. We promote technological neutrality in development. Containerization packages single services and complex applications in autonomous containers, which exhibit great isolation capabilities and are portable across different technological environments. While imposing the platform-as-a-service paradigm, nowadays it seems to preserve the tech neutrality property and to sustain a stepwise migration pattern towards a Statistical SOA. Future work should therefore explore the latter solution due to its promise of simplifying platform configuration, and offering lightweight runtimes, which sustain orchestration, scheduling, scalability and security issues at container level. Likewise, work which further promotes system efficiency could, as proven, have further positive impacts in relation to service reuse and sharing, and further work on data virtualization, by assessing selection rules which interface with a single integration layer, which manages different data sources, may increase the service agility (Karpathiotakis et al. 2015).

Briefly, from the experience in SBR domain, we may highlight that agility and efficiency grant service reuse and sharing in relation to multiple source prioritization and validation tasks. A systematic performance assessment in relation to resource utilization, time behavior and capacity may evaluate efficiency in processing and communication; while rule-based solutions, micro-service patterns, data virtualization and cooperation in development may provide service agility. Future exploration of container-based architectures seems promising in granting other non-functional requirements, such as scalability, security, maintainability and service portability in a technological neutral perspective.

6. References

Alagiannis, I., R. Borovica, M. Branco, S. Idreos, and A. Ailamaki. 2012. "NoDB: Efficient Query Execution on Raw Data Files." In Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data: 241–252. Scottsdale, Arizona, U.S.A. May 20–24, 2012. Doi: http://dx.doi.org/10.1145/2213836.2213864.

Ananthanarayanan, G. 2013. *Optimizing Parallel Job Performance in Data-Intensive Clusters*. Diss. University of California, Technical report Berkeley EECS. Available at: https://www2.eecs.berkeley.edu/Pubs/TechRpts/2014/ (accessed November 2017).

Ananthanarayanan, G., A. Ghodsi, S. Shenker, and I. Stoica. 2013. "Effective Straggler Mitigation: Attack of the Clones." *In NSDI* 13: 185–198. ISBN: 978-1-931971-00-3. Available at https://www.usenix.org/conference/nsdi13/technical-sessions/presentation/ananthanarayanan (accessed November 2017).

Chen, Y. and D.Z. Wang. 2014. "Knowledge Expansion Over Probabilistic Knowledge Bases." In Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data: 649–660. Snowbird, Utah, U.S.A. June 22–27, 2014. Doi: http://dx.doi.org/10.1145/2588555.2610516.

Chen, T., R. Bahsoon, and A.R.H. Tawil. 2014. "Scalable Service-Oriented Replication with Flexible Consistency Guarantee in the Cloud." *Information Sciences* 264: 349–370. Doi: http://dx.doi.org/10.1016/j.ins.2013.11.024.

Cheng, Yu, and F. Rusu. 2015. "Scanraw: A Database Meta-Operator for Parallel In-Situ Processing and Loading." *ACM Transactions on Database Systems (TODS)* 40(3): 19. Doi: http://dx.doi.org/10.1145/2818181.

Delimitrou, C. and C. Kozyrakis. 2014. "Quasar: Resource-Efficient and Qos-Aware Cluster Management." In *ACM SIGPLAN Notices* 49(4): 127–144. ACM. Doi: http://dx.doi.org/10.1145/2644865.2541941.

De Sa, C., A. Ratner, C. Ré, J. Shin, F. Wang, S. Wu, and C. Zhang. 2016. "DeepDive: Declarative Knowledge Base Construction." *ACM SIGMOD Record* 45(1): 60–67. Doi: http://dx.doi.org/10.1145/3060586.

Di Zio, M., N. Fursova, T. Gelsema, S. Giessing, U. Guarnera, J. Petrauskienė, L. Quenselvon Kalben, M. Scanu, K.O. ten Bosch, M. van der Loo, and K. Walsdorfer. 2016. "Methodology for Data Validation." ESSNET ValiDat Foundation. Available at https://ec.europa.eu/eurostat/cros/system/files/methodology_for_data_validation_v1.0_rev-2016-06_final.pdf (accessed November 2017).

Dragoni, N., M. Mazzara, S. Giallorenzo, F. Montesi, A. Lluch Lafuente, R. Mustafin, and L. Safina. 2017. "Microservices: Yesterday, Today, and Tomorrow. In Present and Ulterior Software Engineering." Springer Berlin Heidelberg. Doi: http://dx.doi.org/10.1007/978-3-319-67425-4_12.

ESSnet. 2015. "Enterprise Architecture Reference Framework." Available at https://ec.europa.eu/eurostat/cros/content/ess-enterprise-architecture-reference-framework_en (accessed November 2017).

ESSnet Core Project. 2011. "Common Reference Environment." Available at https://ec.europa.eu/eurostat/cros/content/core_en (accessed November 2017).

ESSnet ValiDat Integration. 2017. "Harmonising Data Validation Approaches in the ESS." Available at https://ec.europa.eu/eurostat/cros/content/essnet-validat-integration_en (accessed November 2017).

Fowler, M. 2014. "A definition of this new architectural term". Available at http://martinfowler.com/articles/microservices.html (accessed November 2017).

Goede, R. 2011. "Agile Data Warehousing: The Suitability of Scrum as Development Methodology." In Proceedings of the 5th IADIS Multi Conference on Computer Science and Information Systems (MCCSIS'2011): 51–58. Rome, Italy. 20–26 July 2011. Available at http://ims.mii.lt/ims/konferenciju_medziaga/MCCSIS/I_WAC_TNS_2011.pdf#page=72 (accessed November 2017).

Gramaglia, L. 2015. "Towards a European Validation Architecture." ESSNET ValiDat Foundation. Available at https://ec.europa.eu/eurostat/cros/content/workshop_en (accessed November 2017).

GSBPMv5.0. 2017. The Generic Statistical Business Process Model. Available at https://statswiki.unece.org/display/GSBPM/GSBPM+v5.0 (accessed November 2017).

GSDEM. 2015. *The Generic Statistical Data Editing Models*. Available at https://statswiki.unece.org/display/sde/GSDEMs (accessed November 2017).

Idreos, S., I. Alagiannis, R. Johnson, and A. Ailamaki. 2011. "Here are my data files. here are my queries. where are my results?" In Proceedings of 5th Biennial Conference on

Innovative Data Systems Research (No. EPFL-CONF-161489). Asilomar, California, U.S.A., January 9–12, 2011. Available at http://cidrdb.org/cidr2011/Papers/CIDR11_Paper7.pdf (accessed November 2017).

Karpathiotakis, M., I. Alagiannis, T. Heinis, M. Branco, and A. Ailamaki. 2015. "Just-In-Time Data Virtualization: Lightweight Data Management with ViDa." In Proceedings of the 7th Biennial Conference on Innovative Data Systems Research (CIDR) (No. EPFL-CONF-203677). Asilomar, California, U.S.A., January 4–7, 2015. Available at https://infoscience.epfl.ch/record/203677/files/vida-cidr.pdf (accessed November 2017).

Karpathiotakis, M., A. Ioannis, and A. Anastasia. 2016. "Fast Queries Over Heterogeneous Data Through Engine Customization." *Proceedings of the VLDB Endowment* 9(12): 972–983. Doi: http://dx.doi.org/10.14778/2994509.2994516.

Khadka, R., A. Saedi, A. Idu, J. Hage, and S. Jansen. 2012. "Legacy to SOA evolution: A systematic literature review. Migrating Legacy Applications": *Challenges in Service Oriented Architecture and Cloud Computing Environments*: 40. Doi: http://dx.doi.org/10.4018/978-1-4666-2488-7.ch003.

Krawatzeck, R., B. Dinter, and D.A. Pham Thi. 2015. "How to make business intelligence agile: The Agile BI actions catalog." In System Sciences (HICSS), 2015 48th Hawaii International Conference on: 4762–4771. 5–8 January 2015. Hawaii, U.S.A. IEEE. Doi: http://dx.doi.org/10.1109/HICSS.2015.566.

Lavrac, N. 2001. "Data Mining and Decision Support: A note on the issues of their integration and their relation to Expert Systems." In the workshop on Integrating Aspects of Data Mining, Decision Support and Meta-Learning IDDM. Available at http://kt.ijs.si/Branax/IDDM-2001_submissions/Lavrac.pdf (accessed November 2017).

Liang, S., P. Fodor, H. Wan, and M. Kifer. 2009. "OpenRuleBench: An analysis of the performance of rule engines." In Proceedings of the 18th international conference on World Wide Web: 601–610. Madrid, Spain. April 20–24, 2009. ACM. Doi: http://dx.doi.org/10.1145/1526709.1526790.

Milani, B.A. and N.J. Navimipour. 2016. "A Comprehensive Review of the Data Replication Techniques in the Cloud Environments: Major Trends and Future Directions." *Journal of Network and Computer Applications* 64: 229–238. Doi: http://dx.doi.org/10.1016/j.jnca.2016.02.005.

Mohamed, M.F. 2016. "Service Replication Taxonomy in Distributed Environments." *Service Oriented Computing and Applications* 10(3): 317–336. Doi: 10.1007/s11761-015-0189-7.

Montoya, G., H. Skaf-Mollia, P. Molli, and M.-E. Vidal. 2017. "Decomposing Federated Queries in Presence of Replicated Fragments." *Web Semantics: Science, Services and Agents on the World Wide Web* 42: 1–18. Doi: http://dx.doi.org/10.1016/j.websem.2016.12.001.

Namiot, D. and M. Sneps-Sneppe. 2014. "On Micro-Services Architecture." *International Journal of Open Information Technologies* 2(9): 24–27. Available at http://injoit.org/index.php/j1/article/view/139 (accessed November 2017).

O'Brien, L., P. Brebner, and J. Gray. 2008. "Business transformation to SOA: aspects of the migration and performance and QoS issues." In Proceedings of the 2nd international

workshop on Systems development in SOA environments: 35–40. Leipzig, Germany. May 10–18, 2008. ACM. Doi: http://dx.doi.org/10.1145/1370916.1370925.

Osrael, J., L. Froihofer, and K.M. Goeschka. 2006. "What Service Replication Middleware Can Learn from Object Replication Middleware." In Proceedings of the 1st workshop on Middleware for Service Oriented Computing (MW4SOC 2006): 18–23. Melbourne, Australia. November 27 – December 01, 2006. ACM. Doi: http://dx.doi.org/10.1145/1169091.1169094.

Prokop, H. 1999. *Cache-oblivious algorithms*. Doctoral dissertation, Massachusetts Institute of Technology. Available at http://supertech.csail.mit.edu/papers/Prokop99.pdf (accessed November 2017).

Pullokkaran, L.J. 2013. *Analysis of Data Virtualization and Enterprise Data Standardization in Business Intelligence*. Doctoral dissertation, Massachusetts Institute of Technology. Available at http://hdl.handle.net/1721.1/90703 (accessed November 2017).

Quensel-von Kalben, L. 2017a. "SERV – Adopting Common Statistical Production Architecture (CSPA) in Europe." *NTTS 2017*. Doi: http://dx.doi.org/10.2901/EUROSTAT.C2017.001.

Quensel-von Kalben, L. 2017b. "Validation, shared services and enterprise architecture: how it fits." *UNECE SDE*. Available at https://www.unece.org/index.php?id=43887 (accessed November 2017).

Razavian, M. and P. Lago. 2015. "A Systematic Literature Review on SOA Migration." *Journal of Software: Evolution and Process* 27(5): 337–372. Doi: http://dx.doi.org/10.1002/smr.1712.

Scannapieco, M., L. Tosco, C. Vaccari, and A. Virgillito. 2011. "A Common Reference Architecture for National Statistical Institutes: the CORA Project." *NTTS 2011*. Doi: http://dx.doi.org/10.2901/Eurostat.C2011.001.

Schafer, M. 2015. A study on VTL. *A Study on the Validation and Transformation Language*. Available at https://ec.europa.eu/eurostat/cros/content/essnet-validation-study-vtl-final_en (accessed November 2017).

Stodder, D. 2013. *Achieving Greater Agility with Business Intelligence*. TDWI Best Practices Report, First Quarter. Available at http://info.attivio.com/rs/attivio/images/TDWI-and-Attivio-Best-Practices-Report-Achieving-Greater-Agility-with-Business-Intelligence-Q1-2013.pdf (accessed November 2017).

Subhlok, J., J.M. Stichnoth, D.R. O'Hallaron, and T. Gross. 1993. "Exploiting Task and Data Parallelism on a Multicomputer." In *ACM SIGPLAN Notices* 28(7): 13–22. ACM. Doi: http://dx.doi.org/10.1145/173284.155334.

Tian, Y., I. Alagiannis, E. Liarou, A. Ailamaki, P. Michiardi, and M. Vukolić. 2017. "DiNoDB: an Interactive-speed Query Engine for Ad-hoc Queries on Temporary Data." *IEEE Transactions on Big Data*. Doi: http://dx.doi.org/10.1109/TBDATA.2016.2637356.

Van Der Lans, R.F. 2013. *Creating an Agile Data Integration Platform using Data Virtualization*. R20 consultancy technical whitepaper. Available at http://stonebond.com/wp-content/uploads/2014/02/Rick-Van-Der-Lans-Whitepaper-May-2013.pdf (accessed November 2017).

Xavier, M.G., M. Neves, F. Rossi, T. Ferreto, T. Lange, and C. de Rose. 2013. "Performance evaluation of container-based virtualization for high performance

computing environments. Parallel, Distributed and Network-Based Processing (PDP)." 2013 21st Euromicro International Conference on. IEEE. Belfast, United Kingdom, 27 Februari–1 March 2013. Doi: http://dx.doi.org/10.1109/PDP.2013.41.

Xie, G., G. Zeng, Y. Chen, Y. Bai, Z. Zhou, R. Li, and K. Li. 2017. "Minimizing Redundancy to Satisfy Reliability Requirement for a Parallel Application on Heterogeneous Service-oriented Systems." *IEEE Transactions on Services Computing*. Doi: http://dx.doi.org/10.1109/TSC.2017.2665552.

Yu, S., C. Wang, K. Ren, and W. Lou. 2010. "Achieving Secure, Scalable, and Fine-Grained Data Access Control in Cloud Computing." *In Infocom, 2010 proceedings IEEE*: 1–9. Ieee. Doi: http://dx.doi.org/10.1109/INFCOM.2010.5462174.

Zhou, X., Y. Chen, and D.Z. Wang. 2016. "ArchimedesOne: Query Processing Over Probabilistic Knowledge Bases." *Proceedings of the VLDB Endowment* 9(13): 1461–1464. Doi: http://dx.doi.org/10.14778/3007263.3007284.

Zissis, D. and D. Lekkas. 2012. "Addressing Cloud Computing Security Issues." *Future Generation Computer Systems* 28(3): 583–592. Doi: http://dx.doi.org/10.1016/j.future.2010.12.006.

Received June 2017
Revised April 2018
Accepted July 2018