Big Data Assurance: An Approach Based on Service-Level Agreements

Claudio A. Ardagna,[1*] Nicola Bena,[1]
Cedric Hebert,[2] Maria Krotsiani,[3]
Christos Kloukinas,[3] George Spanoudakis[3]
[1] Department of Computer Science, Università degli Studi di Milano, Milan, Italy,
[2] SAP Labs France, Mougins, France,
[3] City, University of London, London, UK;
*To whom correspondence should be addressed;
E-mail: claudio.ardagna@unimi.it

January 25, 2023

Keywords: Big Data Assurance, Big Data Analytics, Service-Level Agreements

Abstract: Big Data management is a key enabling factor for enterprises that want to compete in the global market. Data coming from enterprise production processes, if properly analyzed, can provide a boost in the enterprise management and optimization, guaranteeing faster processes, better customer management, and lower overheads/costs. Guaranteeing a proper Big Data pipeline is the Holy Grail of Big Data, often opposed by the difficulty of evaluating the correctness of the Big Data pipeline results. This problem

is even worse when Big Data pipelines are provided as a service in the cloud, and must comply with both laws and users' requirements. To this aim, assurance techniques can complete Big Data pipelines, providing the means to guarantee that they behave correctly, towards the deployment of Big Data pipelines fully compliant with laws and users' requirements. In this paper, we define an assurance solution for Big Data based on Service-Level Agreements, where a semi-automatic approach supports users from the definition of the requirements to the negotiation of the terms regulating the provisioned services, and the continuous refinement thereof.

1 Introduction

Big Data is a major research topic, leading all productive environments and enterprises towards the data-driven economy. Big Data becomes fundamental for all enterprises, from big ones to SMEs, that want to compete in the global market. Better management of data coming from productive processes in fact leads to faster processes, better customer management, and lower overheads and costs. Often, the immense benefits of Big Data are forbidden to those enterprises that do not have in-house Big Data skills and competences, and are therefore unable to manage the intrinsic complexity of such technologies.

To this aim, the R&D community has focused in the last few years on widening the adoption of Big Data technologies by providing solutions supporting users in easily implementing a Big Data campaign [9, 21]. Several approaches to Big Data-as-a-Service have been defined, providing services for the management and execution of Big Data pipelines at different layers of the cloud stack. Substantial work has been done on Big Data Platform-as-a-Service, where users are provided with a pre-configured platform, as for instance Microsoft Azure

HDInsight and Amazon EMR. Users need to only concentrate on configuring and executing the analytics without worrying about how to manage and deploy the corresponding platform. This scenario, however, collides with the lack of data scientists having the skills and competences to implement a sound Big Data campaign and retrieve meaningful results.

Following this issue, different techniques supporting the concept of Big Data Analytics-as-a-Service have been defined [5, 42, 6, 24, 13], where high-level requirements of the users are transformed in Big Data workflows that can be executed on the target Big Data platform. However, these approaches suffer from the inability to evaluate and manage the quality and correctness of the implemented Big Data analytics. It is therefore increasingly important to guarantee that the overall Big Data infrastructure complies with users' expectation, and even more with national/international laws and regulations. This challenge points to the concept of *Big Data assurance*, which aims to provide justifiable confidence that a Big Data pipeline behaves as expected. In the past, assurance techniques (i.e., audit, certification, compliance) have been used to evaluate the status of distributed systems such as the cloud [4]. These solutions, however, barely apply to Big Data scenarios and require a rethinking of the assurance concept at large. Effectively tackling such issue is fundamental to increase trust in the Big Data-as-a-Service paradigm, and in turn to foster the movement of critical businesses to it.

The approach in this paper enables the definition and execution of trustworthy Big Data pipelines with provable compliance to non-functional (e.g., security) properties. It enriches the Model-based Big Data Analytics-as-a-Service (MBDAaaS) methodology in [7] with the notion of *assurance*, by the means of generic annotations. The proposed assurance methodology is integrated with Service-Level Agreements (SLAs), which help negotiating the Big Data services

to execute the pipeline, and evaluating its compliance to non-functional properties, towards full support of users' requirements.

The contribution of this paper is twofold. We first refine the MBDAaaS methodology by introducing assurance requirements and controls at declarative and procedural levels using annotations. We then integrate and take advantage of SLAs management, by *i)* supporting users in SLAs negotiation by automatically generating SLAs negotiation rules according to the provided requirements; *ii)* defining an integrated approach that continuously refine the negotiated services towards full compliance.

The remainder of this paper is organized as follows. Section 2 presents related work and our reference scenario. Section 3 introduces the MBDAaaS methodology, then enriched with assurance support in Section 4. Section 5 details the proposed SLA specification, while Section 6 describes the integration of SLA lifecycle within the MBDAaaS methodology. Section 7 presents a walkthrough of the whole methodology. Finally, Section 8 draws our conclusions.

2 Related Work and Reference Scenario

2.1 Related Work

Security Assurance. It is be defined as a way to gain justifiable confidence that a system will consistently demonstrate one or more non-functional (e.g., security, privacy, and dependability) properties, and operationally behaves as expected despite failures and attacks [4]. It provides methodologies supporting different properties based on several techniques [4], such as *i)* audit [31, 8]; *ii)* certification/labelling [3, 36, 2]; *iii)* compliance [11]. At a lower level, they typically rely on *i)* testing [3, 2]; *ii)* monitoring [11, 31]; *iii)* hybrid and other approaches (e.g., testing and risk [36]). These techniques support different sys-

tems, from (hybrid) cloud environments [11, 31, 3, 2], to edge computing [8], and Internet of Things [36].

SLAs. They are contracts between service providers and consumers. Most of all, SLAs are used to specify the terms for such service provision. In general, research on SLAs followed the evolution of the ICT landscape. The first languages aimed at web services, starting from Web Service Level Agreement [32] and other formalisms [30] culminated in WS-Agreement [1] and corresponding refinements and extensions [39, 28]. Then, other languages emerged addressing the requirements of cloud computing [19] (e.g., CSLA [29], SLAC [46]). Finally, novel languages and paradigms are under investigation in the context of cloud-IoT architectures (e.g., Multi-Level SLA [20]). In general, these languages aim to simplify the SLA specification process for the involved parties, and provide an adequate level of expressiveness. The research community adopted SLAs also for other purposes, such as service [26, 43] and cloud provider selection [40]. Despite the extended research on SLAs, they still fail to completely address the requirements of Big Data Analytics services, as two main issues remain mostly unsupported: *i)* security and privacy requirements; *ii)* complex services and operations dependencies, that are typical of these environments.

Assessment of Big Data Pipelines. It is an active line of research for which many solutions have been proposed. Some of them are based on SLAs [52], despite the aforementioned issues. They mainly target MapReduce jobs [48, 23, 50], aiming to identify and address SLA violations. Other approaches focus on anomaly detection in Big Data pipelines [17, 41, 45, 53], and specific security properties such as integrity [51], availability [27], and privacy [37, 49]. Finally, another flourishing research line specifically addresses *quality* assurance [25], that partially overlaps with security assurance as they both consider non-functional properties.

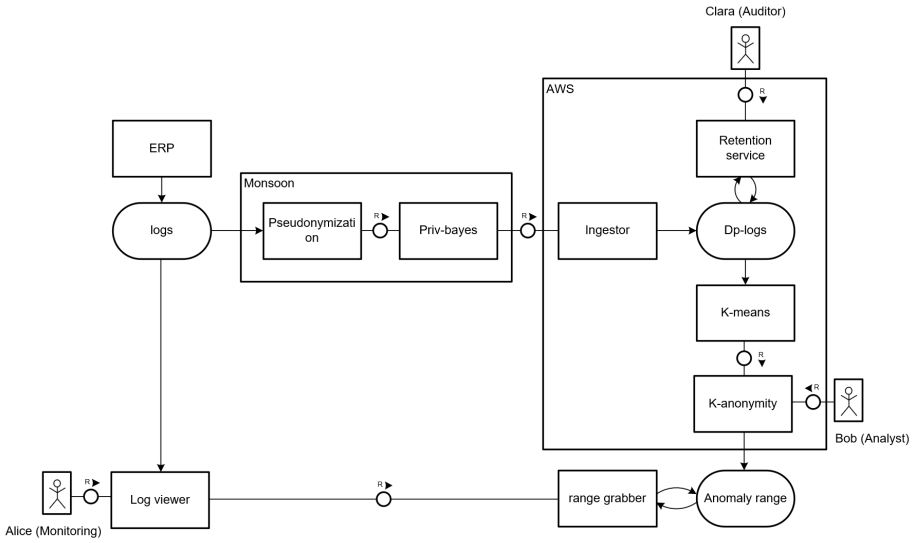

Figure 1: Architecture diagram of our reference scenario

Following the state of the art, Big Data assurance is a pressing need, as it is a key enabler for trustworthy Big Data analytics; SLAs specification and management must be also part of the system, as they are the formal agreements between service consumers and providers. The approach in this paper provides a first boost in this direction, taking the best out of the combination of assurance and SLAs.

2.2 Reference Scenario

Our reference scenario in Figure 1 is provided by SAP, a global market leader in enterprise resource planning, where an ERP system continuously sends its application logs to two connected cloud platforms, a private cloud Monsoon instance and a publicly-accessible AWS instance saving data into a Hadoop data store. In the private cloud, the data are first pseudonymized and then anonymized with differential privacy before being streamed to the AWS instance.

There, the data are constantly monitored by a Retention Service, configured to delete logs older than 1 year. An auditor (Clara) can connect to this service and manually check whether there is a violation of the retention policy. An analyst (Bob) can check the logs through an analytics agent, which first clusters logs and then further anonymize clusters with k-anonymity. k-anonymity is configured to run only on large clusters, as small clusters are considered anomalies, hence not eligible for k-anonymity. Once Bob completes his analysis, he defines anomaly ranges which allow the response team (Alice) to look at the raw logs whitelisted by Bob, enforcing a 4-eyes principle. Alice uses a log viewer to take the proper decision, such as contacting the identified fraudster for a face-to-face interview. When using the log viewer, the raw, non-anonymized logs are displayed to her, but only if they pertain to the ranges defined in the anomaly range storage by Bob (e.g., specific log IDs, specific time-ranges).

3 MBDAaaS Methodology

Big Data analytics and management are practical and pressing needs. While a Big Data pipeline can be easily deployed, it still takes substantial expert knowledge to set the pipeline up in such a way that it actually produces meaningful and sound results. Current approaches mostly focus on providing the so-called Big Data Platform-as-a-Service (BDPaaS) paradigm, where Big Data providers (e.g., Azure, Amazon) offer Big Data platforms on demand to end users. End users then access a complete Big Data platform, without the need of knowing how to install and configure it, and just focus on implementing the Big Data computation. BDPaaS is fundamental to support the users in the management of Big Data analytics and in all involved activities. Though important, however, it is insufficient to widen the adoption of Big Data technologies among the different production domains.

The most recent trends in Big Data application development point to a model-based approach, where users specify their expectations in terms of requirements, and users/consultants follow them in implementing the Big Data pipeline [5]. The aim is to reduce as much as possible the involvement of the users in Big Data management. In the following, we briefly summarize such an approach.

3.1 MBDAaaS Models and Workflow

Unlike existing solutions focusing mostly on data modeling and representation [33], the proposal in this paper is based on the approach in [5], permitting to define declarative requirements along all phases of a Big Data pipeline: *i) data preparation*, consisting of all activities to be done at data collection/ingestion time to prepare data for analytics (e.g., cleansing and anonymization); *ii) data representation*, consisting in the choices for representing data to be analyzed (e.g., data model, data structure, and data management); *iii) data analytics*, specifying the mining operations to execute (e.g., the type and the model of analytics, the learning approach); *iv) data processing*, specifying data routing and parallelization (e.g., real time, batch); *v) data visualization and reporting*, specifying the representation of the outcome of the pipeline (e.g., information on the dimensionality, cardinality).

We develop on the approach in [5], which uses three models and different model transformations, as follows.

Declarative model. It collects user's requirements and expectations on the Big Data pipeline. The user accesses a graphical interface and defines such requirements in terms of *goals*. Formally, a goal is modeled as a triple $\langle g, i, o \rangle$, where g is the name of the goal; i is the *indicator* representing the technique to measure/assess the goal; and o is the target to achieve to consider the goal

fulfilled. Each goal can also be enriched with constraints as pairs (*attribute*, *value*), further refining user's expectations. The declarative model is expressed as a JSON file, and represents the input of the MBDAaaS methodology.

Procedural model. It is a technology-independent model that defines the workflow of the Big Data pipeline. It is a direct acyclic graph where *i)* each node represents a service in one of the five phases of a Big Data pipeline; *ii)* each arc between two nodes represents the execution flow. Each service is configured according to constraints in the declarative model, or preferences specified directly by the user. The procedural model defines a service composition specified in a semantic language based on OWL-S [35].

Deployment model. It is a technology-dependent model that specifies the execution of the Big Data pipeline on the target platform. It is a representation of the procedural model using a workflow language (e.g., Oozie, Spring Cloud Data Flow) that can be automatically executed on the target platform. A compiler transforms the OWL-S procedural model in a language-specific workflow. The deployment model contains all details on the target system and implemented algorithms.

Transformations. They are a series of semi-automatic model transformations that take as input the declarative model and produce as output the deployment model. On the basis of the user-defined goals and constraints in the declarative model, our methodology retrieves the set of compatible services. Then, the user manually configures and composes a subset of them to produce the procedural model. This model is then automatically transformed by a compiler in a deployment model, which is ready to be executed on the target platform [5].

Example 1. *Following the reference scenario in Section 2.2, Listing 1, Figure 2 and Listing 2 present an example of declarative, procedural, and deployment*

Listing 1: Declarative model of reference scenario

```
{
   // [...]
   "tdm:targetDataSources": [
      "hdfs://192.168.0.5:8020"
   ],
   // [...]
   "tdm:processing": {
      //  [...]
      "@type": "tdm:Phase",
      "tdm:label": "Data Processing",
      "tdm:incorporates": [
         {
            "@type": "tdm:Goal",
            "tdm:label": "Anonymization",
            "tdm:constraint": "{}",
            "tdm:incorporates": [
               {
                  "@type": "tdm:Indicator",
                  "tdm:label": "Anonymization Techniques",
                  "tdm:constraint": "{}",
                  "tdm:visualisationType": "Option",
                  "tdm:incorporates": [
                     {
                        "@type": "tdm:Objective",
                        "tdm:constraint": "{k:10}",
                        "tdm:label": "K-Anonymity"
                     }
                  ]
               }
            ]
         }
      ]
   }
   // [...]
}
```

models respectively. SAP defines the declarative model in Listing 1, including the following goals.

G1 Anonymization *for the data preparation phase, with indicator* Anonymization technique *and value* Pseudonymization*, extended with a constraint* (Algorithm, hash=SHA-256).

G2 Anonymization *for the data preparation phase, with indicator* Anonymization technique *and value* Priv-bayes *[47], extended with a constraint on the noise level* (Algorithm, ϵ=0.1).

G3 Compliance *for the data analytics phase, with indicator* Data Erasure *and value* 1 year.

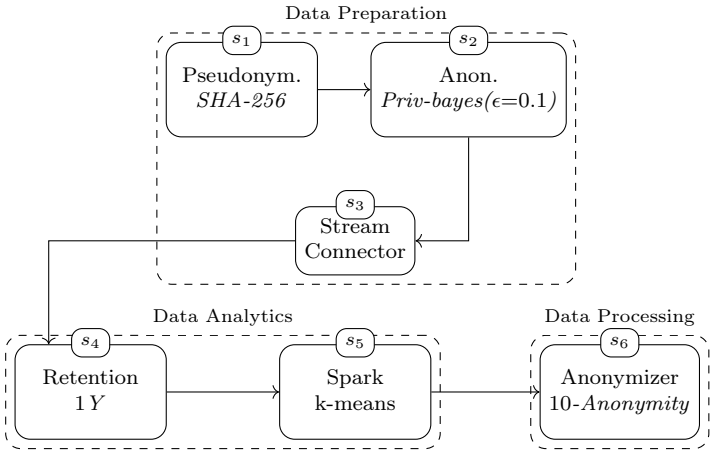

Figure 2: Procedural model of reference scenario

Listing 2: Deployment model of reference scenario

```
# PSEUDONYMIZATION
pseudonymization-service
--parameter.output_column=pseudo
--parameter.input_column=user
--parameter.input_file=hdfs://user/root/sap/anon1/input.csv
--parameter.input_data=hdfs://user/root/sap/anon1/input_data.csv
--parameter.delimiter=;
--parameter.output_file=hdfs://user/root/sap/anon1/output.csv
# ANONYMIZATION
privbayes-service
--input_file=hdfs://user/root/sap/anon1/output.csv
--delimiter=;
--output_file=hdfs://user/root/sap/anon1/results.csv
```

G4 Analytics *for the data analytics phase, with* i) *indicator* Task *and value* Crisp Clustering*; and* ii) *indicator* Learning Approach *and value* Unsupervised.

G5 Anonymization *for the data processing phase, with indicator* Anonymization technique *and value* k-anonymity*, extended with two constraints:* i) *on the value of* k (Algorithm, k=10)*;* ii) *on the cluster size, prescribing to anonymize only large clusters* (Exception, Cluster-size<5).

Our methodology then retrieves all compatible services, leading to a proce-

dural model which applies i) *pseudonymization and* ii) *anonymization;* iii) *a stream data connector to ingest data;* iv) *a retention service deleting old logs;* v) *an analytics service clustering logs;* vi) *an anonymization service generalizing the content before visualization. Finally, the user configures the above services (and the composition thereof), and the compiler generates the deployment model in the Spring Cloud Data Flow language, including all the details of the composition.*

4 Big Data Assurance

The success of a methodology that assists users in instantiating their Big Data pipelines depends on its ability to guarantee that the retrieved pipelines results are in line with user's expectations. In our case, the retrieved results must be the results of a pipeline correctly implementing the user's declarative specifications. The soundness of this entire process depends on whether the instantiated Big Data pipeline satisfies the assurance requirements specified by the user.

4.1 Assurance Requirements

Assurance requirements on Big Data pipelines express user's expectations at both functional and non-functional levels. These requirements are later translated into real assurance controls at deployment level (Section 4.2). With no lack of generality, an assurance requirement is modeled as an assurance annotation, as follows.

Definition 4.1. *An assurance annotation is a triple* ⟨attr op value⟩*, where* attr *is an assurance target;* op *is an operator* $\in\{=, <, >, \neq, \leq, \geq\}$*; and* value *is the objective for the attribute.*

We note that an assurance annotation can also be specified as a logical combi-

nation of other annotations. Our definition captures different ways of expressing assurance annotations, including SLAs, certification, audit, and compliance. It can be easily mapped to different formalisms and (higher) level of abstractions, such as fuzzy logic [18, 44, 14] or specific languages [12, 10, 38, 15], where *value* in the triple $\langle attr\ op\ value \rangle$ can be replaced with words increasing the usability for end users. We note that each annotation can also specify the *importance* a specific non-functional requirement plays in the pipeline for the user, such as, for instance, *desirable* or *mandatory*. In the following, according to the state of the art [3], we assume all annotations to be *mandatory* and leave the specific treatment and balancing of users' non-functional requirements for our future work. These annotations can be specified at two levels, as follows.

Declarative level defines assurance annotations on specific goals of the declarative model in Section 3.1. These annotations are then refined and integrated in a semi-automatic manner during the transformation in the procedural model.

Example 2. *Following Example 1, goal* G5 *is annotated with* $\langle k{=}10 \rangle$*, prescribing the usage of k-anonymity with* $k{=}10$*.*

Procedural level defines assurance annotations on specific services of the procedural model in Section 3.1. These annotations can be the refinement of annotations at declarative level or be directly specified by the user.

Example 3. *Following Example 2, the service* s_6 *implementing k-anonymity can be annotated with* $\langle location{=}EU \rangle$*, prescribing to perform the computation in Europe.*

4.2 Assurance Process

The MBDAaaS assurance process based on the above annotations consists of three steps, as sketched in Figure 3 and described in the following.

Requirements Definition. It extends declarative and procedural models with annotations specifying assurance requirements according to any specific formalisms compatible with Definition 4.1. We note that some assurance requirements can be automatically derived during the transformation at procedural level. At this step, the user refines all the annotations, each of them associated with the corresponding services.

Weaving. It weaves the procedural model and the corresponding annotations into a deployment model, following the approach in Section 3. It transforms assurance annotations into *assurance controls*, that is, assurance scripts returning a Boolean result, indicating the success ($\top$) or failure ($\bot$) of the corresponding annotation. Each control is also associated with a scheduling rule specifying when it should be executed (e.g., after the corresponding services complete their execution, at pre-defined intervals). In other words, the deployment model *i)* includes a ready-to-be-executed workflow of the pipeline and *ii)* integrates all assurance controls to evaluate the compliance between the pipeline and the user's expectations.

Execution and Refinement. It executes the deployment model to perform the Big Data computation. It includes an *assurance manager* managing the life cycle of the assurance controls according to the specified scheduling rules. Finally, the retrieved assurance results can guide the refinement of the declarative model towards full compliance with the user's expectations.

4.3 Multi-Layer Evaluation

Our Big Data assurance methodology extends traditional assurance to embrace different layers of evaluation: platform, analytics process, and data. In particular, the latter points to *veracity* of the 5V model for Big Data [16], referring to data trustworthiness. These three layers map to the five phases of a Big Data

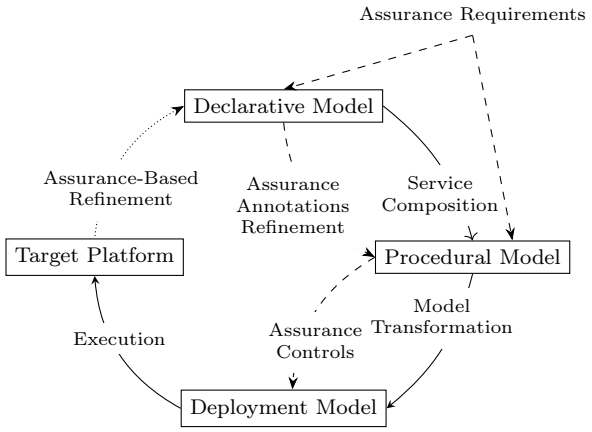

Figure 3: Integration of MBDAaaS methodology and assurance.

pipeline (data preparation, data representation, data analytics, data processing, data visualization and reporting in Section 3.1), as summarized in Table 1 and described in the following.

Data assurance. It evaluates how data are managed, represented, and prepared for the Big Data platform. It relates to phases data preparation and data representation. In our scenario, the analyst must never see individual data records, but only data buckets generalized by applying k-anonymity. To this aim, an assurance control verifies that each displayed data record are indistinguishable from at least $k-1$ other records.

Process assurance. It evaluates the status of a Big Data pipeline by monitoring all the activities between the ingestion of data and the production of the pipeline outcome. It relates to phases data analytics, data visualization and reporting. In our scenario, data must not be kept on the cloud for more than one year. To this aim, an assurance control verifies that data records are never older than one year.

Platform assurance. It evaluates the status of the Big Data platform. It goes

Table 1: Mapping between pipeline phases and assurance layers

Assurance Layer	**MBDAaaS Phases**	**Description**
Data	Data preparation, Data representation	Evaluation of non-functional properties of data
Process	Data analytics, Data visualization & reporting	Evaluation and monitoring of non-functional properties of Big Data pipeline, results, and visualization
Platform	Data processing	Evaluation of non-functional properties of platform components, their deployments and configurations

beyond phase data processing and also considers the components supporting non-functional requirements such data security, and performance. In our scenario, the Big Data platform must guarantee data availability by replicating data on several nodes. To this aim, an assurance control verifies that Hadoop correctly replicates data.

In the remaining of the paper, we describe the integration of the proposed assurance methodology and SLAs management. First, we detail the SLA specification (Section 5), second, we show the overall integration (Section 6).

5 SLA Definition

Typically, Big Data pipelines are executed by service providers. Hence, the users need to negotiate with the providers to define the SLAs specifying the assurance guarantees on the provisioned Big Data services. SLAs are formal agreements (i.e., contracts) that clarify the service provisioning and the responsibilities between the service consumer and the service provider [52]. At a minimal level, SLAs define the non-functional properties to preserve during the provision of a service, and the penalties to apply in case of violations. Our SLA specification language extends WS-Agreement to cover the MBDAaaS concept. Figure 4 shows the three main sections composing an SLA in WS-Agreement. The first section, *Name*, provides an optional SLA name. The second section, *Context*, contains metadata for the entire SLA (e.g., participants, lifetime). The third

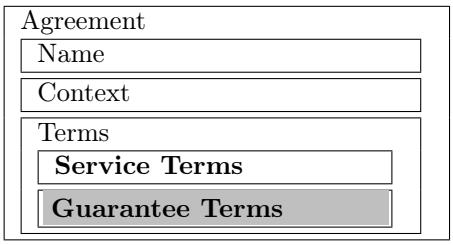

Figure 4: Structure of WS-Agreement SLA

section, *Terms*, specifies the terms of the SLA. They are of two types: *a) Service Terms* specifying the services regulated by the SLA; and *b) Guarantee Terms* specifying the service levels that should be satisfied during the service provision. More in detail, *Guarantee Terms* define the expected quality of service in terms of: *i) ServiceLevelObjective* (SLO) specifying the conditions to be met to fulfill the guarantee, either as Key Performance Indicators (KPIs) or as custom expressions; *ii)* corresponding *BusinessValueList*, specifying different aspects of the SLO, such as importance, penalty, and other custom aspects (*CustomBusinessValue*). Notwithstanding these features, WS-Agreement is ill-suited for our scenario because of the lack of *i)* security and privacy SLOs; *ii)* actions used to react to changes during the SLA life cycle; *iii)* multi-party SLAs, to express Big Data pipelines composed of multiple parties.

For these reasons, we propose an extension of WS-Agreement, supporting actions to undertake when guarantees are violated. To this aim, we extend the sub-element *CustomBusinessValue* of element *BusinessValueList*, being created for this scope [1]. Our extension introduces a new element *CounterActions*, containing one or more elements *CounterAction*, specifying the counteractions to perform if some conditions are met. In particular, each *CounterAction* has an attribute named *Target* specifying the action to undertake. The possible values are: *i) RENEGOTIATION*, causing a SLA negotiation; *ii) PENALTY*, causing a penalty; *iii) REWARD*, causing a reward. A *CounterAction* is guarded by

a condition (sub-element *QualifyingCondition*), that, in turn, consists of one or more atomic conditions (sub-element *Condition*) combined together with a logical operator (e.g., *All* for $\wedge$, *Any* for $\vee$). A *Condition* consists of a comparison operator specified in the attribute *Name*. The operator is applied to two arguments specified in the sub-element *Argument*. In turn, element *Argument* consists of two sub-elements specifying the actual arguments of the condition, either as simple values (*ArgumentValue*) or as a result of a function call (*ArgumentFunction*).

In addition, we provide three built-in functions: *i) VIOLATIONS(GuaranteeTerm)*, returning the number of times the *GuaranteeTerm* has been violated so far; *ii) PENALTY_AMOUNT(GuaranteeTerm)*, returning the amount accrued in penalties due to violations of the *GuaranteeTerm*; *iii) COUNTER(Action, GuaranteeTerm)*, returning the number of times *Action* has been executed following a *GuaranteeTerm* triggering. We note that the *GuaranteeTerm* input of these functions is implicit, as they can only be called from within a *GuaranteeTerm*.

We note that WS-Agreement already supports the specification of penalties, but our extension permits a more precise definition.

Example 4. *Following Example 3, Listing 3 shows an example excerpt of an extended WS-Agreement, starting from element* CustomBusinessValue*. Namespace* wsga *refers to the base WS-Agreement, while* wsga-x *refers to our extension. It specifies to renegotiate the SLA (lines* 2–21*) if the number of violations (lines* 4–18*) is* $\geq$5 *(lines* 6–16*).*

6 SLA-Based Big Data Assurance

Our MBDAaaS platform acts as a middleman between service consumers and providers in a multi-party platform, where some consumers and providers are

Listing 3: Example of WS-Agreement extension

```
<wsag:CustomBusinessValue>
  <wsag-x:CounterActions>
    <wsga-x:CounterAction Target="RENEGOTIATION">
      <wsag:QualifyingCondition>
        <wsag:All>
          <wsag-x:Condition
            Name="GREAT-OR-EQUAL">
            <wsag-x:Argument>
              <wsag-x:ArgumentFunction>
                <wsag-x:Function Name="VIOLATIONS"/>
              </wsag-x:ArgumentFunction>
              <wsag-x:ArgumentValue>
                5
              </wsag-x:ArgumentValue>
            </wsag-x:Argument>
          </wsag-x:Condition>
        </wsag:All>
      </wsag:QualifyingCondition>
    </wsga-x:CounterAction>
  </wsag-x:CounterActions>
</wsag:CustomBusinessValue>
</wsag:BusinessValueList>
```

already enrolled, that is, their corresponding SLAs have been already negotiated when a new party arrives (i.e., consumer). Here, the goal is to let the new party agree on a SLA compatible with all the others (i.e., providers). In general, the SLA lifecycle is a five-steps process [19]: *i) negotiation*, where the different parties negotiate the SLA; *ii) establishment*, where the requested services are deployed; *iii) monitoring*, where the SLA guarantees are monitored ensuring compliance; *iv) violation management*, where eventual violations are properly addressed; *v) reporting and termination*, where a report about the SLA is produced, and the SLA is terminated according to the defined termination rules. The SLA-based Big Data assurance methodology in this paper closely follows this lifecycle, and consists of two main phases: *i) models definition* (including negotiation, establishment); *ii) SLAs monitoring* (including monitoring, violation management, reporting and termination); we describe them in the following and summarize this mapping in Table 2.

Table 2: Mapping between the phases of SLA lifecycle and the phases of our methodology

Our methodology	**SLA lifecycle**
Models definition	Negotiation, establishment
Monitoring	Monitoring, violation management, reporting and termination

6.1 Models Definition

The user first defines the declarative model in Section 3 together with the assurance annotations, which are then refined and paired with additional annotations in the procedural model; a compiler finally produces the deployment model detailing the assurance controls corresponding to the assurance annotations. We note that the compiler specifies the fine-grained scheduling rules to execute the controls (e.g., immediately following the execution of the corresponding service). The defined models are then the input of a second compiler which generates the *negotiation rules* to follow to negotiate the final SLAs with the providers. More in detail, the rules model the interactions forming the negotiation process between the new incoming party and the other ones for which SLAs have been already negotiated. This compiler translates all the concepts in the assurance-enriched models in negotiation rules, and, through negotiation, to the SLAs and, finally, to the provisioned services. We note that assurance annotation importance in Section 4.1 is removed from the provisioned Big Data platform, since it is used for negotiation only.

Our approach is based on the PROSDIN negotiation framework, a proactive run-time SLA negotiation tool [34]. PROSDIN acts as a *negotiation broker* executing negotiation rules on behalf of the interested parties. We automatically derive negotiation rules in the form of XML rules from the models in Section 4. These rules are of the form:

IF (condition) **THEN** (effect) **ELSE** (effect)

Conditions are either atomic conditions or logical combinations of atomic conditions over specific property attributes of the relevant services. Effects can be of three types: *i) accept*, accepting the value of a (set of) property attribute in a SLA offer; *ii) reject*, rejecting the value of a (set of) property attribute in a SLA offer; *iii) set*, proposing a new value or range of values for a (set of) property attribute as part of an SLA offer. Section 7 presents an example of negotiation rules. PROSDIN then translates these rules into *Jess rules*, that is, the rule engine it uses internally [22].

In the simpler approach, we derive *matching rules only*, that is, the MBDAaaS platform tries *only* to match the new SLA with the terms of the SLAs of the other parties, with no counteroffers being made (that is, no effect *set*). Otherwise, we can derive *full negotiation rules*, where the MBDAaaS platform tries to negotiate the terms of the new SLA to obtain a match. In case the terms do not match, a counteroffer is made proposing new values (using effect *set*). We note that other negotiation approaches exist, as described in [7].

Finally, once the negotiation process is over, our approach produces the corresponding SLA(s) in the extended WS-Agreement specification (Section 5). Then, the SLA(s) is weaved together with the deployment model, injecting the (optional) actions upon violation.

6.2 Monitoring

The Big Data platform includes the corresponding assurance controls, which are executed according to the fine-grained scheduling rules specified during weaving (Section 4.2). This continuous execution enables to monitor the compliance to the negotiated SLAs resulting from the generated negotiation rules and, in turn, to the assurance requirements. More in detail, every time an assurance control fails (i.e., it returns $\perp$), it signals a non-compliance of the service it is associ-

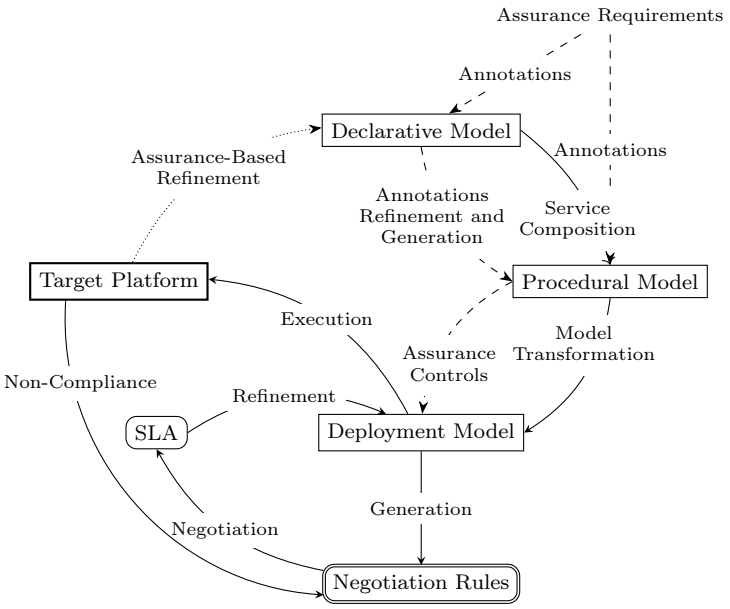

Figure 5: Overview of the whole SLA-based assurance process

ated with (Section 4.2). To the aim of providing a (mostly) automated solution, our assurance methodology goes beyond merely detecting non-compliance, by proposing and possibly applying remediations encoded in the generated SLAs. In fact, thanks to the WS-Agreement extension in Section 5, (counter-)actions, guards, and SLA functions are directly specified in the SLA and integrated with the scheduling rules of the assurance manager. For instance, action guards are translated into low-level conditions such as conditions on the results of assurance controls. This way, when a non-compliance is detected, it can trigger either a SLA renegotiation or a penalty. Figure 5 sketches the overall process. We highlight that if renegotiation actions are used, our methodology keeps monitoring and negotiating new SLAs until non-compliance events are no longer detected. In other words, it permits to *converge* towards a service provision that matches all user's requirements.

Table 3: Assurance annotations

#	Goal	Annotations	Type	Description
D1	*G1*	⟨|Out|=256⟩	A	Check output length
D2	*G2*	⟨ϵ=0.1⟩	A	Check noise level
D3	*G3*	⟨Age≤1Y⟩	A	Check retention
D4	*G4*	⟨k=2⟩	M	Check number of clusters (benign and malign)
D5	*G5*	⟨k=10⟩ ∧ ⟨|Cluster| ≥ 5⟩	M	Check k-anonymity

(a) declarative

#	Serv.	Annotations	Type	Description
P1	s_3	⟨size(In)=size(Out)⟩	M	All data are ingested
P2	s_3	⟨location=EU⟩	M	Computation in EU
P3	s_4	⟨location=EU⟩	M	Computation in EU
P4	s_5	⟨location=EU⟩	M	Computation in EU
P5	s_6	⟨location=EU⟩	M	Computation in EU

(b) procedural

7 Walkthrough Example

Following the reference scenario in Section 2.2 and Examples 1–4, we provide a full run of our methodology.

The first step involves the definition of the models in Section 3 detailing: *i)* the goals of the Big Data pipeline in the declarative model (Example 1), consisting of two anonymization goals (*G1*, *G2*) in the data preparation phase, one-year retention (*G3*) and clustering (*G4*) in the data analytics phase, and anonymization (*G5*) in the data processing phase; *ii)* the service composition realizing such goals in the procedural model (Figure 2), built by automatically selecting compatible services and manually composing and configuring them.

This step also involves the definition of assurance requirements, as annotations on the declarative and procedural models above (Section 4). During the production of the procedural model, some requirements are *i)* added automatically and *ii)* translated into annotations in the resulting procedural model. More in detail, Table 3(a) shows the assurance annotations at declarative level, distinguishing between those specified by users ("M") and those derived automatically ("A"). Table 3(b) shows the assurance annotations at procedural level;

Table 4: Assurance controls and corresponding annotations

#	**Annotations**	**Schedule**
C1	*P1*	After
C2–C5	*P2–P5*	During
C6	*D3* refined at procedural-level	During

for brevity, we omitted those refining annotations at declarative level.

Next, a compiler weaves together the above models and annotations producing the deployment model by transforming *i)* the abstract service composition of the procedural model into a concrete service composition, in this case in Spring Cloud Data Flow language; *ii)* the procedural-level assurance annotations into assurance controls, as part of the concrete service composition. We note that these transformations are semi-automatic, that is, the user is also involved in the refinement of the abstract service composition. Table 4 shows an excerpt of procedural-level assurance annotations and corresponding assurance controls. Some of them are executed by the assurance manager when the corresponding services complete their execution (e.g., *C1*), others run in pair (e.g., *C2–C5*, *C6*).

At this point, the service consumer still need to retrieve the AWS services she is going to use. In other words, the consumer needs to negotiate the SLAs. Our methodology supports her also in this step. It automatically produces SLA negotiation rules from the above models, then executed by PROSDIN (Section 6.1). In this case, we consider the simpler negotiation approach, trying only to match with the provider's SLA.

Listing 4 shows an excerpt of the generated negotiation rules. They consist of two conditions chained with a logical AND (lines 3–22); they indicate that if a service provider has made an (counter-)offer where *i)* the price to pay for penalties is <100€ (lines 4–11), and *ii)* the pseudonymization algorithm is *SHA-256* (lines 13–21), then those values should be accepted (lines 24–33). Once

Listing 4: Jess rule for negotiation.

```
<tnsr:NegotiationRule name="rule1">
   <tnsr:If>
      <tnsr:LogicalExpression>
         <slac:Condition relation="LESS-THAN">
            <slac:Arg1><slac:QualityAttribute
             name="PENALTY-AMOUNT" party="PROVIDER"/>
            </slac:Arg1>
            <slac:Arg2><slac:Constant
             type="NUMERICAL" unit="EUR"> 100
            </slac:Constant></slac:Arg2>
         </slac:Condition>
         <slac:LogicalOperator> AND </slac:LogicalOperator>
         <slac:Condition relation="EQUALS">
            <slac:Arg1><slac:QualityAttribute
              name="PSEUDONYMIZATION" party="PROVIDER"
              unit="SHA-256"/>
            </slac:Arg1>
            <slac:Arg2><slac:Constant
               type="BOOLEAN">TRUE</slac:Constant>
            </slac:Arg2>
         </slac:Condition>
      </tnsr:LogicalExpression>
   </tnsr:If>
   <tnsr:Then>
      <tnsr:Action>
         <tnsr:Accept>
            <tnsr:QualityAttribute
               name="PENALTY-AMOUNT" party="PROVIDER"/>
            <tnsr:QualityAttribute
               name="PSEUDONYMIZATION" party="PROVIDER"/>
         </tnsr:Accept>
      </tnsr:Action>
   </tnsr:Then>
</tnsr:NegotiationRule>
```

the negotiation is over, an SLA is produced according to the specification in Section 5; Listing 5 shows an excerpt thereof. The SLA has id *55* (line 1), is called *SAP-Log-Analytics* (line 5), and is referred to service s_6 (line 15). It extends Example 4, prescribing a renegotiation after 5 violations (lines 37–53) and a penalty of 75€ (lines 54–60). The negotiated values for the anonymization algorithm are contained within element *CustomServiceLevel* (lines 17–31), while the amount to pay for penalty within element *Penalty* (lines 55–58). In addition, it includes native support for assurance annotation importance (see Section 4.1) by translating in the WS-Agreement construct *Importance*. In our case, value 3 refers to importance *mandatory* (line 34).

Our methodology refines the deployment model, by injecting the generated

actions into the model; they are mapped into low-level functionalities of the platform and integrated with existing assurance controls and corresponding scheduling rules. When a SLA renegotiation is triggered, the negotiation starts once again, leading to an improved deployment model. This process is repeated, ensuring that the resulting deployment model is compliant to user's expectations. We recall that such expectations are expressed as assurance annotations from which negotiation rules are generated.

8 Conclusions

We presented a methodology that aims to unleash a trustworthy MBDAaaS paradigm, where assurance annotations, either generated automatically or specified by users, continuously assess the behavior of Big Data pipelines. The models in Section 3 and 4 are the basis to generate negotiation rules that finally lead to the SLAs regulating the services provisioned for the pipeline. The deployed platform is continuously monitored according to the rules defined in the negotiated SLA, and is continuously improved towards full compliance to users' expectations.

Acknowledgments

Research supported, in parts, by EC H2020 Project CONCORDIA GA 830927 and Università degli Studi di Milano under the program "Piano sostegno alla ricerca".

References

[1] A. Andrieux, K. Czajkowski, K. Keahey, A. Dan, K. Keahey, H. Ludwig, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu. Web services agreement specification (WS-Agreement). In *Global Grid Forum GRAAP-WG*, Honolulu, HI, USA, June 2004.

[2] M. Anisetti, C. A. Ardagna, N. Bena, and E. Damiani. Stay Thrifty, Stay Secure: A VPN-Based Assurance Framework for Hybrid Systems. In *Proc. of SECRYPT 2020*, Paris, France, July 2020.

[3] M. Anisetti, C. A. Ardagna, E. Damiani, and F. Gaudenzi. A Semi-Automatic and Trustworthy Scheme for Continuous Cloud Service Certification. *IEEE TSC*, 13(1), January–February 2020.

[4] C. A. Ardagna, R. Asal, E. Damiani, and Q.H. Vu. From Security to Assurance in the Cloud: A Survey. *ACM CSUR*, 48(1), August 2015.

[5] C. A. Ardagna, V. Bellandi, M. Bezzi, P. Ceravolo, E. Damiani, and C. Hebert. Model-Based Big Data Analytics-as-a-Service: Take Big Data to the Next Level. *IEEE TSC*, 14(2), 2021.

[6] C. A. Ardagna, V. Bellandi, P. Ceravolo, E. Damiani, and R. Finazzo. A Methodology for Cross-Platform, Event-Driven Big Data Analytics-as-a-Service. In *Proc. of IEEE Big Data 2019*, Los Angeles, CA, USA, December 2019.

[7] C. A. Ardagna, E. Damiani, M. Krotsiani, C. Kloukinas, and G. Spanoudakis. Big Data Assurance Evaluation: An SLA-Based Approach. In *Proc. of IEEE SCC 2018*, San Francisco, CA, USA, July 2018.

[8] M. Aslam, B. Mohsin, A. Nasir, and S. Raza. FoNAC - An automated Fog Node Audit and Certification scheme. *COSE*, 93, June 2020.

[9] M. D. Assunção, R. N. Calheiros, S. Bianchi, M. A. S. Netto, and R. Buyya. Big Data computing and clouds: Trends and future directions. *Journal of Parallel and Distributed Computing*, 79, 2015.

[10] Z. Azmeh, M. Driss, F. Hamoui, M. Huchard, N. Moha, and C. Tibermacine. Selection of Composable Web Services Driven by User Requirements. In *Proc. of IEEE ICWS 2011*, Washington, DC, USA, July 2011.

[11] A. Ciuffoletti. Application level interface for a cloud monitoring service. *Computer Standards & Interfaces*, 46, 2016.

[12] G. Damiano, E. Giallonardo, and E. Zimeo. onQoS-QL: A Query Language for QoS-Based Service Selection and Ranking. In *Proc. of ICSOC 2007 Workshops*, Vienna, Austria, September 2007.

[13] Databricks. *Unified Analytics*. https://databricks.com/.

[14] S. De Capitani di Vimercati, S. Foresti, G. Livraga, V. Piuri, and P. Samarati. A Fuzzy-based Brokering Service for Cloud Plan Selection. *IEEE ISJ*, 13(4), December 2019.

[15] S. De Capitani di Vimercati, S. Foresti, G. Livraga, V. Piuri, and P. Samarati. Supporting User Requirements and Preferences in Cloud Plan Selection. *IEEE TSC*, 14(1), January–Feburary 2021.

[16] Y. Demchenko, P. Membrey, P. Grosso, and C. de Laat. Addressing Big Data Issues in Scientific Data Infrastructure. In *Proc. of CTS 2013*, San Diego, CA, USA, May 2013.

[17] M. Elsayed and M. Zulkernine. Towards Security Monitoring for Cloud Analytic Applications. In *Proc. of IEEE BigDataSecurity, HPSC, IDS 2018*, Omaha, NE, USA, May 2018.

[18] C. Esposito, M. Ficco, F. Palmieri, and A. Castiglione. Smart Cloud Storage Service Selection Based on Fuzzy Logic, Theory of Evidence and Game Theory. *IEEE Transactions on Computers*, 65(8), August 2016.

[19] F. Faniyi and R. Bahsoon. A Systematic Review of Service Level Management in the Cloud. *ACM CSUR*, 48(3), December 2015.

[20] P. Grubitzsch, I. Braun, H. Fichtl, T. Springer, T. Hara, and A. Schill. ML-SLA: Multi-Level Service Level Agreements for Highly Flexible IoT Services. In *Proc. of IEEE ICIOT 2017*, Honolulu, HI, USA, June 2017.

[21] I. Hashem, I. Yaqoob, N. Anuar, S. Mokhtar, A. Gani, and S. Khan. The rise of "big data" on cloud computing: Review and open research issues. *Information Systems*, 47, 2015.

[22] E. F. Hill. *Jess in Action: Java Rule-Based Systems*. Manning Publications Co., Greenwich, CT, USA, 2003.

[23] E. Hwang and K. H. Kim. Minimizing Cost of Virtual Machines for Deadline-Constrained MapReduce Applications in the Cloud. In *Proc. of ACM/IEEE GRID 2012*, Beijing, China, September 2012.

[24] Improvado. https://improvado.io/.

[25] S. Ji, Q. Li, W. Cao, P. Zhang, and H. Muccini. Quality Assurance Technologies of Big Data Applications: A Systematic Literature Review. *Applied Sciences*, 10(22), 2020.

[26] F. Jrad, J. Tao, A. Streit, R. Knapper, and C. Flath. A utility-based approach for customised cloud service selection. *International Journal of Computational Science and Engineering*, 10, 2015.

[27] Z. Ke and N. Park. Availability Modeling and Assurance of Map-Reduce Computing. In *Proc. of IEEE DASC/PiCom/DataCom/CyberSciTech 2017*, Orlando, FL, USA, November 2017.

[28] K. T. Kearney, F. Torelli, and C. Kotsokalis. SLA*: An Abstract Syntax for Service Level Agreements. In *Proc. of ACM/IEEE Grid 2010*, Brussels, Belgium, October 2010.

[29] Y. Kouki, F. A. D. Oliveira, S. Dupont, and T. Ledoux. A Language Support for Cloud Elasticity Management. In *Proc. of IEEE/ACM CCGrid 2014*, Chicago, IL, USA, May 2014.

[30] D. D. Lamanna, J. Skene, and W. Emmerich. SLAng: A Language for Defining Service Level Agreements. In *Proc. of FTDCS 2003*, San Juan, Puerto Rico, USA, May 2003.

[31] S. Lins, S. Schneider, and A. Sunyaev. Trust is Good, Control is Better: Creating Secure Clouds by Continuous Auditing. *IEEE TCC*, 6(3), 2018.

[32] H. Ludwig, A. Keller, A. Dan, R. P. King, and R. Franck. Web Service Level Agreement (WSLA) Language Specification. Technical report, 2003.

[33] S. Madden. From Databases to Big Data. *IEEE Internet Computing*, 16(3), 2012.

[34] K. Mahbub and G. Spanoudakis. Proactive SLA Negotiation for Service Based Systems: Initial Implementation and Evaluation Experience. In *Proc. of IEEE SCC 2011*, Washington, DC, USA, July 2011.

[35] D. Martin, M. Paolucci, S. McIlraith, M. Burnstein, D. McDermott, D. McGuinness, B. Parsia, T. R. Payne, M. Sabou, M. Solanki, et al. Bringing Semantics to Web Services: The OWL-S Approach. In *Proc. of SWSWPC 2004*, San Diego, CA, USA, July 2004.

[36] S. N. Matheu-García, J. L. Hernández-Ramos, A. F. Skarmeta, and G. Baldini. Risk-based automated assessment and testing for the cybersecurity certification and labelling of IoT devices. *Computer Standards & Interfaces*, 62, February 2019.

[37] A. Mehmood, I. Natgunanathan, Y. Xiang, G. Hua, and S. Guo. Protection of Big Data Privacy. *IEEE Access*, 4, 2016.

[38] O. Moser, F. Rosenberg, and S. Dustdar. Domain-Specific Service Selection for Composite Services. *IEEE TSE*, 38(4), July–August 2012.

[39] S. Nepal, J. Zic, and S. Chen. WSLA+: Web Service Level Agreement Language for Collaborations. In *Proc. of SCC 2008*, Honolulu, HI, USA, July 2008.

[40] C. Redl, I. Breskovic, I. Brandic, and S. Dustdar. Automatic SLA Matching and Provider Selection in Grid and Cloud Computing Markets. In *Proc. of ACM/IEEE Grid 2012*, Beijing, China, September 2012.

[41] L. Rettig, M. Khayati, P. Cudré-Mauroux, and M. Piórkowski. Online Anomaly Detection over Big Data Streams. In *Applied Data Science*. Springer, 2019.

[42] E. R. Sparks, S. Venkataraman, T. Kaftan, M. J. Franklin, and B. Recht. KeystoneML: Optimizing Pipelines for Large-Scale Advanced Analytics. In *Proc. of IEEE ICDE 2017*, San Diego, CA, USA, April 2017.

[43] A. Taha, S. Manzoor, and N. Suri. SLA-Based Service Selection for Multi-Cloud Environments. In *Proc. of IEEE EDGE 2017*, Honolulu, HI, USA, September 2017.

[44] A. Taha, R. Trapero, J. Luna, and N. Suri. A Framework for Ranking

Cloud Security Services. In *Proc. of IEEE SCC 2017*, Honolulu, HI, USA, September 2017.

[45] S. Thudumu, P. Branch, J. Jin, and J. Singh. A comprehensive survey of anomaly detection techniques for high dimensional big data. *Journal of Big Data*, 7(1), Jul 2020.

[46] R. B. Uriarte, F. Tiezzi, and R. De Nicola. SLAC: A Formal Service-Level-Agreement Language for Cloud Computing. In *Proc. of the IEEE/ACM UCC 2014*, London, UK, December 2014.

[47] J. Vaidya, B. Shafiq, A. Basu, and Y. Hong. Differentially private naive bayes classification. In *Proc. of IEEE/WIC/ACM WI and IAT 2013*, Atlanta, GA, USA, November 2013.

[48] A. Verma, L. Cherkasova, and R. H. Campbell. Aria: Automatic resource inference and allocation for mapreduce environments. In *Proc. of ACM ICAC 2011*, Karlsruhe, Germany, June 2011.

[49] L. Wang, J. P. Near, N. Somani, P. Gao, A. Low, D. Dao, and D. Song. Data Capsule: A New Paradigm for Automatic Compliance with Data Privacy Regulations. In *Heterogeneous Data Management, Polystores, and Analytics for Healthcare*. 2019.

[50] Y. Wang and W. Shi. On Scheduling Algorithms for MapReduce Jobs in Heterogeneous Clouds with Budget Constraints. In *Proc. of OPODIS 2013*, Hong Kong, China, December 2013.

[51] Y. Wang, J. Wei, M. Srivatsa, Y. Duan, and W. Du. IntegrityMR: Integrity Assurance Framework for Big Data Analytics and Management Applications. In *Proc. of IEEE Big Data 2013*, Santa Clara, CA, USA, October 2013.

[52] X. Zeng, S. Garg, M. Barika, A. Y. Zomaya, L. Wang, M. Villari, D. Chen, and R. Ranjan. SLA Management for Big Data Analytical Applications in Clouds: A Taxonomy Study. *ACM CSUR*, 53(3), June 2020.

[53] X. Zhou, Y. Hu, W. Liang, J. Ma, and Q. Jin. Variational LSTM Enhanced Anomaly Detection for Industrial Big Data. *IEEE Transactions on Industrial Informatics*, 17(5), 2021.

Listing 5: Example of negotiated SLA in extended WS-Agreement

```
<wsag:Agreement AgreementId="55"
  xmlns:wsag="http://www.ggf.org/namespaces/ws-agreement"
  xmlns:wsag-x="https://www.mbdaas.com/namespaces/ws-agreement-ext"
  xmlns:aws-kanon="https://aws-kanon.com/namespace">
  <wsag:Name>SAP-Log-Analytics</wsag:Name>
  <wsag:AgreementContext>
    <!-- ... -->
  </wsag:AgreementContext>
  <wsag:Terms>
    <wsag:All>
      <!-- .... -->
      <wsag:GuaranteeTerm
        Name="ANONYMIZATION-KANON"
        Obligated="ServiceProvider">
        <wsag:ServiceScope ServiceName="S6"/>
        <wsag:ServiceLevelObjective>
          <wsag:CustomServiceLevel>
            <aws-kanon:ServiceDetails>
              <aws-kanon:Function
                Name="K-Anonymization">
                <aws-kanon:Config>
                  <aws-kanon:Parameter
                    Name="K"
                    Value="10"/>
                  <aws-kanon:Parameter
                    Name="MinClusterSize"
                    Value="5"/>
                </aws-kanon:Config>
              </aws-kanon:Function>
            </aws-kanon:ServiceDetails>
          </wsag:CustomServiceLevel>
        </wsag:ServiceLevelObjective>
        <wsag:BusinessValueList>
          <wsag:Importance>3</wsag:Importance>
          <wsag:CustomBusinessValue>
            <wsag-x:CounterActions>
              <wsga-x:CounterAction Target="RENEGOTIATION">
                <wsag:QualifyingCondition>
                  <wsag:All>
                    <wsag-x:Condition
                      Name="GREAT-OR-EQUAL">
                      <wsag-x:Argument>
                        <wsag-x:ArgumentFunction>
                          <wsag-x:Function Name="VIOLATIONS"/>
                        </wsag-x:ArgumentFunction>
                        <wsag-x:ArgumentValue>
                          5
                        </wsag-x:ArgumentValue>
                      </wsag-x:Argument>
                    </wsag-x:Condition>
                  </wsag:All>
                </wsag:QualifyingCondition>
              </wsga-x:CounterAction>
              <wsga-x:CounterAction Target="PENALTY">
                <wsag-x:wsag-x:Penalty>
                  <wsag:ValueUnit>EUR</wsag:ValueUnit>
                  <wsag:ValueExpression>75</wsag:ValueExpression>
                </wsag-x:wsag-x:Penalty>
                <!-- then, same as RENEGOTIATION -->
              </wsga-x:CounterAction>
            </wsag-x:CounterActions>
          </wsag:CustomBusinessValue>
        </wsag:BusinessValueList>
      </wsag:GuaranteeTerm>
    </wsag:All>
  </wsag:Terms>
</wsag:Agreement>
```