A Model-Driven Methodology for Big Data Analytics-as-a-Service

Claudio A. Ardagna, Valerio Bellandi, Paolo Ceravolo
Computer Science Department
Università degli Studi di Milano
Crema, Italy
Email: *name.surname@unimi.it*

Ernesto Damiani
CINI - Consorzio Interuniversitario Nazionale per l'Informatica
Rome, Italy
EBTIC, Khalifa University, Abu Dhabi, UAE
Email: *ernesto.damiani@kustar.ac.ae*

Michele Bezzi, Cedric Hebert
Security Research
SAP Labs France
Sophia Antipolis, France
Email: *name.surname@sap.com*

***Abstract*—The Big Data revolution has promised to build a data-driven ecosystem where better decisions are supported by enhanced analytics and data management. However, critical issues still need to be solved in the road that leads to commodization of Big Data Analytics, such as the management of Big Data complexity and the protection of data security and privacy. In this paper, we focus on the first issue and propose a methodology based on Model Driven Engineering (MDE) that aims to substantially lower the amount of competences needed in the management of a Big Data pipeline and to support automation of Big Data analytics. The proposal is experimentally evaluated in a real-world scenario: the implementation of novel functionality for Threat Detection Systems.**

***Index Terms*—Big Data, Model-Driven Architecture, OWL-S**

I. Introduction

Big Data management and analytics are among the most critical pressing needs for all organisations that want to move their business a step forward. Big Data market is expected to substantially grow in the next years [1] and its technologies are introducing a Copernican revolution in the areas of data storage, processing, and analytics. However, the impact and diffusion of Big Data technologies are lowered, on one side, by complex and not standardised technologies, and, on the other side, by the lack of professional profiles with the necessary background and competence, especially in SMEs.

A recent trend has underlined the relevance of users' requirements and developed the idea that achieving the full potential of Big Data analytics needs to embrace a data model approach [15]. Traditional data modeling, which focused on resolving the complexity of relationships among schema-enabled data [14], has been discarded as no longer applicable to Big Data scenarios. In fact, in addition to data representation, Big Data models should provide a shared specification of the process to manage data resources (including anonymization and privacy-preservation procedures) and of the computations to be done over them. They also need to provide all the information to carry out Big Data analytics over commodity execution platforms.

A practical goal for next-generation Big Data is to provide solutions where end-users define their expectations on goals to be achieved with Big Data analytics, while smarter engines manage and compose solutions to deploy Big Data architectures and carry out the expected analytics.

In this paper, we propose a framework for Model-based Big Data Analytics-as-a-Service (MBDAaaS) [2], which supports customers lacking Big Data expertise in managing big data analytics and deploying a full big data pipeline that addresses their goals. The proposed approach is based on a *declarative model* (Section IV-B), specifying the goals of a given analytics in the form of pairs indicators/objectives. The customers navigate and prioritize indicators through an interface, which can guide her in implementing consistent selections by choosing the objective values of low-priority indicators to maximize the objectives of high-priority indicators. Indicators and objectives are then used to incrementally refine a platform-independent *procedural model* (Section IV-C) specifying how analytics should be carried out in terms of an abstract OWL-S workflow. Procedural models are finally compiled in a ready-to-be-executed *deployment model* (Section IV-D) specifying Big Data platform-dependent configurations and supporting automatic provisioning of computational components and resources. Such transformations are implemented in a methodology providing a semi-automatic process to MBDAaaS (Section V), which is practically evaluated (Section VI) in a real-world scenario (Section III).

II. Related Work

Model-based Big Data Analytics-as-a-Service (MBDAaaS) [2] is interconnecting two trends that are currently shaping the Big Data ecosystem. On one side, Big Data technologies perform the best in on-demand and scalable computing infrastructures. On the other side, the high complexity and side-costs of designing, developing and deploying such infrastructures suggest the adoption of model-driven approaches that foster modularity, reusability, and automatisation of design and implementation tasks.

The first aspect is largely discussed in the literature [22], [6]. Recent research has focused on incorporating Cloud computing negotiation models based on Service Level Agreement (SLA) [21], opening the door to self-configuration [7], and cost models for evaluating alternative Big Data-as-a-Service solutions [18].

The second aspect, was intended by several authors in connection to software engineering methodologies operating at design time [11]. However, other authors have investigated methods that operate at execution time. For example, workflow orchestration frameworks were placed on top of distributed data processing applications [8], model-driven approaches were adopted to configure runtime observability frameworks [10], to adapt the data storage model for improving I/O performances [12], or to design visual analytics [4]. In such a context, where multiple parties are involved in the provisioning, processing, and sharing of heterogeneous data, the conciliation of non-functional requirements related to data protection and information privacy is also a demanding challenge. A compete survey on the different strategies implemented for data protection is available in [20]. For example, the processing procedures of data mining algorithms can be masked, using generalization, suppression, permutation or perturbation [13], or adopting differential privacy [5], giving to the data provider the guarantee that the data collector will not be able to consistently read data.

Although all these aspects were considered in the literature, there is a lack of a comprehensive approach addressing the whole lifecycle of MBDAaaS, from the specification of declarative goals to the deployment of a big data analytics, via the definition of a procedural model representing how the expected analytics should work.

III. Reference Scenario: Threat Detection Systems

Threat detection and prevention in software ecosystems [16] have recently been extended to the application layer. Threat Detection System (TDS)[1] detects potential attacks on the application landscape by gathering and analyzing log data, such as user change logs, security audit logs, remote function call gateway logs, and transaction logs. Logs are usually pre-processed, anonymized, translated into a common format, and analyzed by pattern or anomaly detection algorithms, which can highlight suspicious events. Finally, a detailed investigation is performed by a human expert to decide if a real attack is detected or it is a false positive. However, with the increasing size and complexity of software systems, the volume and diversity of log data are becoming major issues. Customers have in place a large spectrum of different systems and a wide range of data security policies. As a result, including and managing these heterogeneous log files currently need a significant customization effort, especially when they contain sensitive and personal information (e.g., user IDs, IP addresses) coming from logs of multiple customers or are accessed by the third party (e.g., cloud provider) running TDS. Similarly, customers often need different security analysis, depending on the security context, industrial sector, risk management policies, and the analytics functionality need to be often customized too. Accordingly, major challenges for data analysis by TDS systems are related to devising a flexible framework supporting:

- the provisioning of customized analytics and reporting approach for stakeholders;
- data anonymization (at different levels, for different customers);
- a scaling approach for variable data loads;
- a limited effort (i.e., semi-automatic) for the integration of new and diverse log files, which also adapts corresponding analytics;

In this paper, we present how MBDAaaS approach can support the design of a TDS system, or its extension to include novel functionality, and illustrate the results of a realization of this approach (Section VI). For sake of simplicity, we will focus on a simple, but relevant, analytics scenario for TDS: advanced anomaly detection analytics, able to limit the number of false positives in alerts. In fact, since TDS are set to minimize the likelihood that a threat is not detected, they often suffer of a large number of false positives, which result in additional investigation effort for security experts. For example, a typical approach for anomaly detection is using deviations from the normalized mean computed from previous events, using metric as z-score (see [3]) or fixed thresholds. It can be done focusing on a single type of event (e.g., the amount of data sent by a user/system in one day) or a combination of event types (e.g., data sent and received). However, the high variability in human and system activities can lead to a large number of outliers, and more sophisticated approaches are needed, such as automatically detecting specific activities (via clustering) and considering outliers from the clusters as anomaly.

IV. Models for Big Data Analytics-as-a-Service

The methodology proposed in this paper, building on MDE paradigm [17], aims to provide an approach that decouples high-level goals of a Big Data campaign from low-level details of the Big Data architecture. Our methodology defines three models as described in the following of this section: *i)* a *declarative model* representing the computation-independent model, *ii)* a *procedural model* representing the platform-independent model, and *iii)* a *deployment model* representing the platform-dependent model.

A. Structuring the Big Data Pipeline

The systematisation proposed by several authors [6], [19] describes a Big Data process along different conceptual areas that structure the deployment in interdependent pipelines. A complete process can be split in 5 main areas driving the whole MBDAaaS.

- *Data preparation area* specifies all activities aimed to prepare data for analytics. For instance, it defines how to guarantee data owner privacy using anonymization (e.g., hashing, obfuscation), and identifies data cleaning and integration techniques.

[1]We refer to these systems as TDS, to distinguish them from network level intrusion detection systems (called IDS or SIEM) [9]. In addition, we base our description on the SAP Enterprise Threat Detection, but the analysis could be applied to other solutions, including IDS.

- *Data representation area* specifies how data are represented and expresses representation choices for each analysis process. For instance, it defines the data model (e.g., document-oriented, graph-based, relational) and the data structure (e.g., structured, semi-structured, unstructured).
- *Data analytics area* specifies the analytics to be computed. For instance, it defines the expected outcome (e.g., descriptive, prescriptive, predictive, diagnostic), the type of analytics (e.g., cluster, classifier, predictor), and the learning approach (e.g., supervised, unsupervised, semi-supervised).
- *Data processing area* specifies how data are routed and parallelized. For instance, it defines the processing type (e.g., real-time, near real-time, batch) and the elasticity level (e.g. full, bounded, none).
- *Data visualisation and reporting area* specifies an abstract representation of how the results of analytics are organised for display and reporting. For instance, it defines data display type (e.g., composition, order).

B. Declarative Models

Declarative models are platform- and vendor-independent models specifying user goals related to the conceptual areas in Section IV-A. They describe customers' goals expressing the properties a Big Data Campaign (BDC) has to fulfil. On the other side, goals express commitments on service properties made by ICT providers to their customers. A *goal* G (e.g., *data source properties*) is measured by an *indicator* I, which is a label expressing a way to assess the goal (e.g., *consistency*), and an *objective* O, which is a threshold on certain scale, either ordinal or metric (e.g., *weak*, *normal*, or *strong*), for I. A declarative model is defined as a set of prioritised goals annotated with metadata specifying constraints on procedural and deployment models, as formally presented in the following definition.

Definition IV.1 (Declarative Model)**.** *A declarative model $d_i \in \mathcal{D}$ consists of five elements $(a_i, \mathcal{G})$, one for each conceptual area $a_i \in \mathcal{A}$, where $\mathcal{G}$ specifies a set of prioritised goals $G_i = \{(I_i, O_i), \mathcal{C}_i, pr_i\}$, with I_i a label representing an indicator of G_i, O_i the value (objective) of I_i, $\mathcal{C}_i = \{c_1, \ldots, c_n\}$ a set of constraints on I_i driving the configuration of procedural and deployment models, and pr_i the priority of the goal G_i.*

We note that, to support users with different big data competences, each constraint $c_j \in \mathcal{C}_i$ can be either defined by the user or enforced by the selected technological platform. A constraint is defined as a boolean formula of expressions of the form *op*(*attr*,*value*), where *op* is an operator in $\{=, \neq, <, >, \leq, \geq, \in\}$, *attr* represents an attribute referring to a procedural/deployment model, and *value* a (set of) value for the given attribute. For instance, a goal G_i=(`Anonimization_Technique`, *k-anonymity*) on property anonymity may set a constraint c on the cardinality of k. This constraint will be considered when configuring the procedural model or when implementing quality assurance controls for the BDC.

C. Procedural Models

Procedural models are *platform-independent models* that formally and unambiguously describe how analytics should be configured and executed. Procedural models are generated following goals and constraints specified in the declarative models. They provide a workflow in the form of a service orchestration that composes, in an arbitrary way, services falling in the areas presented in Section IV-A, as formally defined below.

Definition IV.2 (Procedural Model)**.** *A procedural model m is a direct acyclic graph $G(V,E,\lambda)$, where a vertex $v_i \in V$ refers to a service (e.g., an algorithm, mechanism, or component) in a specific area* area*(v_i), an edge $(v_i, v_j) \in E$ is annotated with function call f_i to the service represented by v_j, and $\lambda: V \rightarrow SC$ is a labeling function that associates a set $\{cp_1, \ldots, cp_n\} \in SC$ of configuration parameters with each $v_i \in V$.*

We note that a configuration parameter cp_i either derives from constraints c in Definition IV.1 or is given as input by the customers during procedural model definition. For instance, cp_i can specify the cardinality k of a data anonymization based on k-anonymity. We also note that each function call annotating an edge in G triggers a state transition and corresponding mechanism execution. For instance, a common procedural model consists of a three-step, sequential workflow that first prepares data (area data preparation), then runs the analytics (area data analytics), and finally presents the results (area display and reporting). Services are selected and orchestrated within the procedural workflow according to the goals stated in the declarative model. For instance, in case goal G =(`Analytics_Task`, *crisp_clustering*) is defined, only services implementing a crisp clustering procedure (e.g., k-means) will be available for area data analytics.

D. Deployment Models

Deployment models specify how procedural models are instantiated and configured on a target platform (platform-dependent models), and drive analytics execution in real scenarios. A deployment model $\overline{G}$ is an instance of a procedural model G, defined as follows.

Definition IV.3 (Deployment Model m_d)**.** *Let $m = G(V,E,\lambda)$ be a procedural model. A deployment model m_d is a direct acyclic graph $\overline{G}(\overline{V},\overline{E},\lambda)$ where:*

- *each vertex $\overline{v}_i \in \overline{V}$ contains the implementation of the corresponding platform-independent service $v_i \in V$,*
- *each edge $(\overline{v}_i, \overline{v}_j) \in \overline{E}$, corresponding to edge $(v_i, v_j) \in E$, is annotated with the endpoint f_i referring to the platform-dependent implementation of the service represented by $\overline{v}_j$, and*
- *$\lambda: \overline{V} \rightarrow \overline{SC}$ associates a set $\{\overline{cp}_1, \ldots, \overline{cp}_n\} \in \overline{SC}$ of platform-dependent configurations with each $\overline{v}_i \in \overline{V}$.*

We note that a there exists an isomorphism between procedural model m and deployment model m_d.

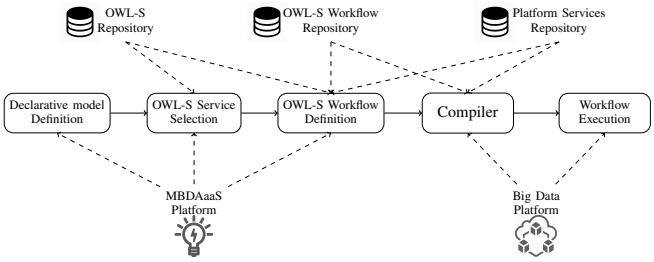

Fig. 1. MBDAaaS Methodology: Execution steps

V. Model-Driven Big Data Analytics

MBDAaaS implements the methodology in Section V-A and is responsible for all activities aimed to configure and execute the Big Data analytics involving: *i) Big Data customer* specifying the goals of its BDC, *ii) Big Data consultant* helping the Big Data customer in specifying all customizations needed to execute her analytics, *iii) MBDAaaS platform* that is responsible for semi-automatically managing and executing a BDC on a *Big Data platform*.

A. MBDAaaS Methodology

MBDAaaS methodology is based on different building blocks which are described in the following.

Declarative Model. It allows customers to define a set of goals shaping a BDC and retrieve a set of services compatible with these goals.

OWL-S Ontology. It specifies the set of abstract services that are available to Big Data customers and consultants for building their BDC. For each service, it defines the interface, a link to the goals in the declarative model, and some constraints driving the definition of a procedural model.

OWL-S Workflow. It is the procedural model providing all artifacts for specifying an abstract Big Data workflow. It defines how relevant OWL-S services can be composed to carry out the Big Data analytics. It supports traditional composition patterns, such as sequence, alternative, parallel.

Platform-Dependent Workflow. It is the platform-dependent version (deployment model) of an abstract OWL-S workflow, which is ready to be executed on the target Big Data platform. he corresponding platform-dependent workflow.

MBDAaaS Compiler. It is the component that takes as input an abstract OWL-S workflow and produces as output the corresponding platform-dependent workflow.

Figure 1 presents the process of our MBDAaaS that is composed of five main steps as follows. In the first step (*Declarative Model Definition*), the Big Data customer produces a declarative model d specifying the goals of a Big Data campaign (see Definition IV.1). In the second step (*OWL-S Service selection*), the OWL-S services compatible with the declarative model specification are selected. We note that service selection is done on the basis of the annotations

```
{
  [...]
  "preparation": {
     "Govern and stewards": {
        "Anonymization Model": "Privacy preserving data publishing",
        "Anonymization Technique": "Hashing",
        "Constraints":
        { "Algorithm": "SHA−256",
             "Target": "UserID" }}
           },
  "analytics": {
     "Analytics Aim": {
        "Models": "Diagnostic",
        "Task": "Clustering",
        "Constraints":
     {"Algorithm": "Crisp Clustering "}, },
      "Analytics Quality": {
        "Positive Predictive Value (Precision)": "Medium",
        "True Positive Rate (Recall, Sensitivity)": "Medium",
        "True Negative Rate (Specificity)": "Medium",
        "False Positive Rate (Fall−out)": "Medium",}
        },
  "processing": {
     "Mode": {
        "Analysis Goal": "Near Real−time",
        "Rounds": "Multiple",
        "Locality": "Distributed"},},
  "display": {
     "Data display": {
        "Processing Type": "Relationship",
        "Dimensionality": "nD",
        "Data Density": "Low",
        "Data Type": "Ratio" },
     "Operations": {
        "Interaction": "Filter",
        "User": "Techie",
        "Goal": "Cluster/Categorize"}
  },
  [...]
}
```

Fig. 2. A fragment of the declarative model in JSON.

to the services in the OWL-S ontology, mapping them to indicators/objectives in the declarative model. In the third step (*OWL-S Workflow Definition*), the Big Data consultant uses the MBDAaaS platform to define the abstract workflow of the Big Data campaign. It is generated by composing the selected OWL-S services and represents the procedural model in Definition IV.2. In the fourth step (*MBDAaaS Compiler*), MBDAaaS platform transforms the OWL-S workflow in a platform-dependent workflow. The latter represents the deployment model in Definition IV.3. This step is crucial to build a semi-automatic MBDAaaS and puts some strong constraints on the generality of the compiler, which needs to adapt to the selected target Big Data platform. Finally, in the fifth step (*Workflow Execution*), MBDAaaS platform executes the analytics on the target Big Data platform.

B. From Declarative to Procedural Models

The methodology in Figure 1 first transforms a declarative model in a procedural model as described in the following.

1) Declarative Model Definition: The process starts with the definition of the declarative model (step 1 in Figure 1), which is presented in this paper using a JSON format. Figure 2 propose an excerpt of a declarative model for areas data preparation, analytics and processing, visualization.

```
<owl:NamedIndividual
 rdf:about="tdm:spark:hashing">
  <rdf:type
   rdf:resource="tdm:MaterialService"/>
  <realizeSpecification
   rdf:resource="tdm:data_preparation:anonymization"/>
</owl:NamedIndividual>
<owl:NamedIndividual
 rdf:about="tdm:data_preparation:anonymization">
  <rdf:type
   rdf:resource="tdm:Anonymization"/>
  <hasArea
   rdf:resource="tdm:dataPreparation"/>
  <hasConstraint
   rdf:resource="tdm:OWLNamedIndividual_00000"/>
  <hasIndicator
   rdf:resource="tdm:data_preparation:anonymization:technique"/>
  <hasObjective
   rdf:resource="tdm:data_preparation:anonymization:technique:hashing"/>
 </owl:NamedIndividual>
<owl:NamedIndividual
 rdf:about="tdm:OWLNamedIndividual_00000">
  <rdf:type rdf:resource="tdm:Constraint"/>
  <hasURIValue rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
    http://toreador−project.eu/example/dataset.csv
  </hasURIValue>
  <onParameter
   rdf:resource="tdm:normalizedDataset"/>
</owl:NamedIndividual>
```

Fig. 3. OWL-S based Hashing Anonymization service profile.

```
<process:AtomicProcess rdf:about="#AnonymizationService:Process">
    <process:hasOutput>
      <process:Output rdf:ID="Dataset">
        <process:parameterType rdf:datatype="http://www.w3.org/2001/
XMLSchema#anyURI">
          &tdm;#AnonimyzedDataset
        </process:parameterType>
      </process:Output>
    </process:hasOutput>
    <service:describes rdf:resource="#AnonymizationService"/>
    <process:hasInput>
      <process:Input rdf:ID="Dataset">
        <process:parameterType rdf:datatype="http://www.w3.org/2001/
XMLSchema#anyURI">
          &tdm;#Dataset
        </process:parameterType>
      </process:Input>
    </process:hasInput>
    <process:hasInput>
      <process:Input rdf:ID="Field">
        <process:parameterType rdf:datatype="http://www.w3.org/2001/
XMLSchema#anyURI">
          &tdm;#Field
        </process:parameterType>
      </process:Input>
    </process:hasInput>
    <rdfs:label>Process</rdfs:label>
  </process:AtomicProcess>
```

Fig. 4. OWL-S based Hashing Anonymization service process.

- *Area preparation*. The *Govern and stewards* goal requires the dataset produced at the previous step to undergo an anonymization process driven by goals anonymization model and anonymization techniques. It requires the adoption of a hashing techniques with constraints on the type of algorithm (i.e., SHA-256) and the target of anonymization (i.e., UserID).
- *Areas Analytics and Processing*. For simplicity, we consider the subset of the goals that drives the transformation from declarative to procedural model: (`Task`, *crisp_ clustering*), (`Analysis Goal`, *near_real_time*), (`True Positive Rate`, *medium*). This portion of the declarative model requires the adoption of a clustering algorithm to be applied in near real time (e.g., micro-batches), with an expected quality measured through true positive rate.
- *Area Visualization and Reporting*. It specifies goals *Data display* and *Operations*, which drive the selection of the visualization approach. This selection considers also specifications in other areas, in particular the need to visualize the results of a clustering analytics.

2) OWL-S Service Selection: The process retrieves those services available in the platform that are compatible with the goals and constraints specified in the declarative model (step 2 in Figure 1), using an extended version of OWL-S. The OWL-S ontology is structured in three interrelated sub-ontologies, known as the *profile*, *process model*, and *grounding*. The *profile* ontology is used to express what "does" the service; the *process model* describes "how it works"; and the *grounding* maps the constructs of the process model onto detailed specifications of message formats, protocols, and so forth. In addition to specifying the service, in our methodology, OWL-S is extended to describe a mapping with the goals of the declarative model, which are used for service selection. Figure 3 shows an example of the OWL-S modeling of SHA-256 hashing service of Spark that is mapped to declarative goals (`Anonymization Model`, *privacy preserving data publishing*) and (`Anonymization Technique`, *hashing*) with constraint *Algorithm: SHA-256*. On the basis of this annotation, SHA-256 hashing service is selected as a service compatible with the specified declarative model. Figure 4 presents the process model of the hashing anonymization service where a dataset is taken as input and the anonymization algorithm is applied to the "field" parameter. The "AnonymizedDataset" is then returned as output. Figure 5 presents an example of grounding, where the function of the anonymization process is specified. In our example, a reference to the WSDL file of the service ("'HashingAnonymizationService.wsdl"') is used as a placeholder that points to the functionality/library provided by the target Big Data platform and implementing the service itself.

Following the above approach, a k-means service is selected for area analytics, a map-reduce parallel processing for area processing, and two visualization techniques, namely, a dimension-reduced plane visualization based on scatter-plot and an independent coordinates visualization based on scatter-plot matrices, for area visualization.

3) OWL-S Workflow Definition: The last step of this process consists in the definition of an OWL-S workflow, representing the procedural model in this paper (step 3 in Figure 1). Here, we present a basic workflow example that composes the Anonymization Service and the Clustering Service (Figure 6).

```
<grounding:WsdlGrounding rdf:about="#Grounding">
  <service:supportedBy rdf:resource="#Service"/>
  <grounding:hasAtomicProcessGrounding>
    <grounding:WsdlAtomicProcessGrounding rdf:ID="AtomicProcessGrounding"/>
  </grounding:hasAtomicProcessGrounding>
</grounding:WsdlGrounding>
<grounding:WsdlAtomicProcessGrounding rdf:about="#AtomicProcessGrounding">
  <grounding:wsdlInput>
    <grounding:WsdlInputMessageMap>
      <grounding:owlsParameter rdf:resource="#Dataset"/>
      <grounding:wsdlMessagePart rdf:datatype="http://...#anyURI">
        file:HashingAnonymizationService.wsdl#Dataset
      </grounding:wsdlMessagePart>
    </grounding:WsdlInputMessageMap>
  </grounding:wsdlInput>
<grounding:wsdlOutput>
  <grounding:WsdlOutputMessageMap>
    <grounding:owlsParameter rdf:resource="#HashingAnonymizedDataset"/>
    <grounding:wsdlMessagePart rdf:datatype="http://...#anyURI">
      file:HashingAnonymizationService.wsdl#anonymizedDataset
    </grounding:wsdlMessagePart>
  </grounding:WsdlOutputMessageMap>
</grounding:wsdlOutput>
[...]
</rdf:RDF>
```

Fig. 5. OWL-S based Hashing Anonymization service grounding.

```
<process:CompositeProcess rdf:about="tdm:CompositeProcess00001">
  <process:composedOf>
    <process:Sequence>
      <process:components>
        <process:ControlConstructList>
          <list:first> <process:Perform>
            <process:process rdf:resource="tdm:AnonymizationService"/>
          </process:Perform> </list:first>
          [...]
        <process:ControlConstructList>
          <list:first> <process:Perform>
            <process:process rdf:resource="tdm:ClusteringService"/>
          </process:Perform> </list:first>
          [...]
        </process:ControlConstructList>
      </process:components>
    </process:Sequence>
  </process:composedOf>
</process:CompositeProcess>
```

Fig. 6. Example of an OWL-S workflow.

C. From Procedural to Deployment Models

The methodology in Figure 1 then transforms the procedural model returned in Section V-B in a deployment model that can be directly executed on the target platform. This transformation is based on a compiler (step 4 in Figure 1) that takes as input the OWL-S workflow returned in step 3 (Figure 6) and information on the target platform (e.g., installed services/algorithms), and produces as output a technology-dependent workflow. This is achieved using a stylesheet-based transformation. We note that, in case no workflow engine is available on the target platform, the compiler returns the specific calls to the services in the platform matching activities in the OWL-S workflow.

Figure 7 shows an example of technology-dependent workflow corresponding to the OWL-S workflow in Figure 6 and compatible with the Oozie workflow engine. The workflow in Figure 7 is composed of three activities in a sequence that execute *i)* SHA-256 anonymization, *ii)* k-means analytics,

```
<workflow-app xmlns = "uri:oozie:workflow:0.4" name = "CompositeProcess00001">
  [...]
  <!Step preparation -->
  <action name="Preparation:Anonymization">
    <spark xmlns="uri:oozie:spark-action:0.1">
      <job-tracker>toreador-project.eu:8088</job-tracker>
      <name-node>node1:8020</name-node>
      <configuration>
        <property>
          <name>mapred.compress.map.output</name>
          <value>true</value>
        </property>
        <property>
          <name>field</name>
          <value>UserID</value>
        </property>
      </configuration>
      <master>local[*]</master>
      <mode>client</mode>
      <name>Spark anonymization</name>
      <class>org.apache.spark.anonymization</class>
      <jar>/lib/spark-sha256.jar</jar>
      <arg>inputpath=hdfs://localhost/input/anonymization/dataset.csv</arg>
    </spark>
    <ok to="Analytics:KMean"/>
    <error to="kill_job"/>
  </action>

  <!Step analysis -->
  <action name="Analytics:KMeans">
    <spark xmlns="uri:oozie:spark-action:0.1">
      <job-tracker>toreador-project.eu:8088</job-tracker>
      <name-node>node1:8022</name-node>
      <configuration>
        <property>
          <name>mapred.compress.map.output</name>
          <value>true</value>
        </property>
        <property>
          <name>K</name>
          <value>$p1</value>
        </property>
        <property>
          <name>minInteraction</name>
          <value>$p2</value>
        </property>
      </configuration>
      <master>local[*]</master>
      <mode>client<mode>
      <name>Spark kmeans</name>
      <class>org.apache.spark.kmeans</class>
      <jar>/lib/spark-kmeans.jar</jar>
      <arg>inputpath=hdfs://localhost/input/kmeans/dataset.csv</arg>
    </spark>
    <ok to = "visualization:notebook" />
    <error to = "kill_job" />
  </action>
      [...]
  <kill name = "kill_job">
    <message>Job failed</message>
  </kill>
  <end name = "end" />
</workflow-app>
```

Fig. 7. Example of an Oozie workflow.

iii) scatter-plot visualization.There is however a subtlety to consider: the workflow in Figure 7 is not completely specified. In fact, the scripts in the workflow refers to templates that need to be instantiated with parameters in the OWL-S workflow.[2] For instance, let us consider the reference to the k-means script */lib/spark-kmeans.jar* in Figure 7. The corresponding script is completely specified unless the variables associated with the parameters number of clusters ($p1) and numbers of iterations

[2]In case some parameters are missing, they are asked to the users.

($p2) in the oozie workflow. When $p1 and $p2 are filled with real values, the workflow is completely specified and ready to be executed.

VI. Workflow Execution: TDS Scenario

The methodology in Figure 1 finally executes the deployment model returned in step 4 on the target big data platform. In the following, we show the realization of the anomaly detection TDS use case (Section III). We focused on devising an anomaly detection functionality for TDS, with a smaller false positive rate compared to traditional systems based on z-score; in particular, we wanted to detect possible anomalies in the amount of data transmitted by users within a software landscape, which may indicate malicious behavior. We used a log data set containing around 12 weeks of activity collected from SAP systems deployed in a test environment. We considered three features, extracted from the log files: Data Sent, Data Received, Data Exchanged (all of them in bytes), aggregated on a duration of 6 hours and per user. For testing purpose, we artificially generated a data set with multi-variate Gaussian distribution having the same average and covariance matrix as the original data points.

We considered two target platforms for execution of the analytics process that resemble the scenarios presented in Section V-C. The first platform is an open Big Data platform (*platform 1*) installing Apache components (e.g., Hadoop YARN, Hbase, Spark, Zeppelin) and including the Oozie workflow engine. The second platform is the SAP proprietary Big Data platform (*platform 2*), installing HANA, XS Engine, APL, and Jupyter, but with no workflow engine on board. The process execution on platform 1 simply required to execute the Oozie workflow in Figure 7, while the one on platform 2 required to manually generate and execute all service calls. Automatic generation of service calls will be the topic of our future work.

The process orchestrated the three steps in Figure 7:

1) *Data preparation:* log dataset was processed to address data minimization privacy requirement removing the unnecessary personal information. Specifically, the UserID values were pseudo-anonymized by a hashing function.
2) *Data processing and analytics*, a k-means algorithm was applied to our data points. For simplicity, here, we focused on the key elements only. In the actual implementation, we also considered additional data processing steps, such as a logarithmic transformation of the data, to reduce skewness, scaling (dividing for the standard deviation and subtracting the mean) and optimization of the k value using the *elbow method*.
3) *Data visualization*, data were visualized as scatter-plot, supporting visual inspection of data by expert users, to select valid alerts.

For space restriction, in Figure 8, we only present the results retrieved through our MBDAaaS using platform 2. Data processing was realized by the APL HANA library, ran on top of the HANA XS engine; data visualization was done using the Jupyter (notebook) environment. Our experiments

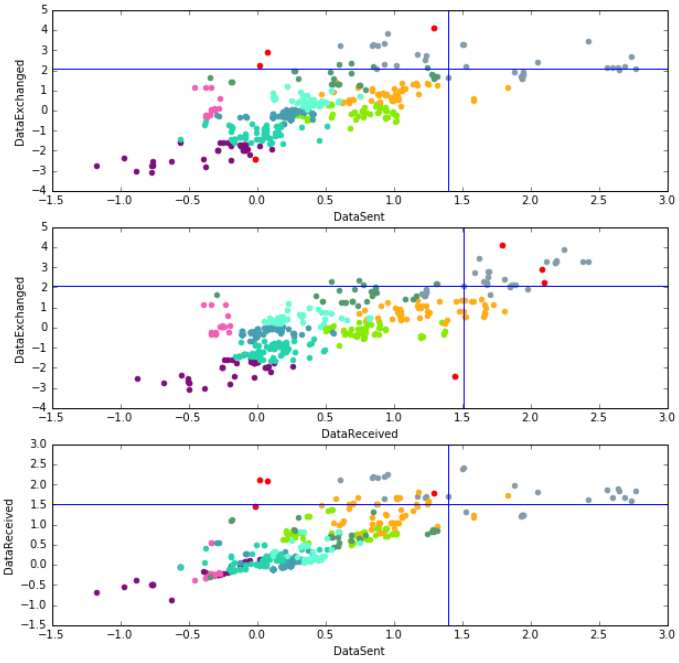

Fig. 8. 2 dimensional projections of (a subset of) data points using SAP platform (platform 2).

returned 6 clusters, representing similar events, with silhouette values around 0.5. Red points indicate *anomalous* data points (outliers) where the distance to the centroid is greater than 99^{th} percentile of training data; colored data points represent the 6 different clusters; blue lines indicate the value of thresholds computed with the z-score method (using z-values optimized for each dimension). We can observe that the number of identified outliers is reduced compared to z-score method (see the number of red dots versus the points above the thresholds). Our experiments show the suitability of a MBDAaaS methodology, where Big Data analytics are automated with a little involvement of customers that only define goals in a declarative model and are possibly queried for additional parameters of the analytics.

VII. Conclusions

This paper proposed a model for managing a MBDAaaS platform and demonstrated its feasibility in addressing a TDS scenario. Evolving from this point we plan to enrich the set of services supported in our platform to offer an environment, insisting on vertical scenarios, for testing real-life BDC. Users will be requested to deal with the different design stages typically addressed in preparing a BDC, to then compare alternative options and investigating the consequences of their choices.

Acknowledgement

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the TOREADOR project, grant agreement No 688797.

REFERENCES

[1] Nasscom Crisil GR&A Analysis. Big data - the next big thing, 2012.

[2] C.A. Ardagna, P. Ceravolo, and E. Damiani. Big data analytics as-a-service: Issues and challenges. In *Proc. of PSBD 2016*, Washington, VA, USA, December 2016.

[3] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM CSUR*, 41(3):15:1–15:58, July 2009.

[4] S. Cheng, B. Wang, W. Zhong, C. Xie, S. Mahmood, J. Wang, and K. Mueller. Model-driven visual analytics for big data. In *Proc. of NYSDS 2016*, New York, USA, August 2016.

[5] C. Dwork and J. Lei. Differential privacy and robust statistics. In *Proc. of STOC 2009*, Washington, USA, May 2009.

[6] 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*, 47:98–115, 2015.

[7] H. Herodotou, H. Lim, G. Luo, N. Borisov, L. Dong, F. Cetin, and S. Babu. Starfish: A self-tuning system for big data analytics. In *Cidr*, volume 11, pages 261–272, 2011.

[8] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. Joseph, R. Katz, S. Shenker, and I. Stoica. Mesos: A platform for fine-grained resource sharing in the data center. In *Proc. of NSDI 2011*, Boston, USA, March 2011.

[9] M. Kaempfer. http://tinyurl.com/mhl3lay, 2015.

[10] J. Klein, I. Gorton, L. Alhmoud, J. Gao, C. Gemici, R. Kapoor, P. Nair, and V. Saravagi. Model-driven observability for big data storage. In *Proc. of WICSA 2016*, Venice, Italy, April 2016.

[11] V. Kumar and P. Alencar. Software engineering for big data projects: Domains, methodologies and gaps. In *Proc. of Big Data 2016*, Washington, USA, December 2016.

[12] J. Liu, S. Byna, B. Dong, K. Wu, and Y. Chen. Model-driven data layout selection for improving read performance. In *Proc. of IPDPSW 2014*, Phoenix, USA, May 2014.

[13] R. Lu, X. Lin, and X. Shen. Spoc: A secure and privacy-preserving opportunistic computing framework for mobile-healthcare emergency. *IEEE TPDS*, 24(3):614–624, 2013.

[14] S. Madden. From databases to big data. *IEEE Internet Computing*, 16(3):4–6, 2012.

[15] V. Markl. Breaking the chains: On declarative data analysis and data independence in the big data era. *Proc. of VLDB Endowment*, 7(13):1730–1733, August 2014.

[16] A. Oprea, Z. Li, T.-F. Yen, S. Chin, and S. Alrwais. Detection of early-stage enterprise infection by mining large-scale log data. In *Proc. of DNS 2015*, Rio de Janeiro, Brazil, June 2015.

[17] D. Schmidt. Guest editor's introduction: Model-driven engineering. *Computer*, 39(2):25–31, February 2006.

[18] G. Skourletopoulos, C. Mavromoustakis, P. Chatzimisios, G. Mastorakis, E. Pallis, and J. Batalla. Towards the evaluation of a big data-as-a-service model: a decision theoretic approach. In *Proc. of INFOCOMW 2016*, San Francisco, USA, April 2016.

[19] D. Terzi, R. Terzi, and S. Sagiroglu. A survey on security and privacy issues in big data. In *Proc. of ICITST 2015*, London, UK, December 2015.

[20] L. Xu, C. Jiang, J. Wang, J. Yuan, and Y. Ren. Information security in big data: privacy and data mining. *IEEE Access*, 2:1149–1176, 2014.

[21] Y. Zhao, R. Calheiros, J. Bailey, and R Sinnott. Sla-based profit optimization for resource management of big data analytics-as-a-service platforms in cloud computing environments. In *Proc. of Big Data 2016*, Washington, USA, 2016. December.

[22] Z. Zheng, J. Zhu, and M. Lyu. Service-generated big data and big data-as-a-service: an overview. In *Proc. of BigData Congress 2013*, Santa Clara, USA, June 2013.