Chapter 1:
A Data Utility Model for Data Intensive Applications in Fog Environments

Cinzia Cappiello, Pierluigi Plebani, Monica Vitali

Politecnico di Milano – Dipartimento di Elettronica ed Informazione
Piazza Leonardo da Vinci 32, 20133 Milano, Italy

Abstract: Sensors, smart devices, and wearables have been widely adopted in recent years, bringing to the production of a big amount of data which can be shared between several applications using these data as input for their analysis. Data intensive applications can get advantage of these data only if they are reliable and if they fit the requirements of the application. Designing data intensive applications requires a trade-off between the value obtained by the analysis of the data, which is affected by their quality and volume, and the performance of the analysis, affected by delays in accessing the data and availability of the data source. In this chapter, we present a *Data Utility model* used to assess the fitness of a data source with respect to its usage in a data intensive application running in a Fog environment. In this context, data are provided using a Data as a Service approach and, both data storage and data processing, can be placed in a cloud resource as well as in an edge device. The placement of a resource affects the quality of the service and the Data Utility as well. On this basis, the Data Utility model provides a support for making decisions on the deployment of data intensive applications according to the impact of the task location, and on the selection of proper data sources as input for the application according to the application requirements, taking into consideration that both tasks and data can be moved from the edge to the cloud, and vice versa, to improve the efficiency and the effectiveness of applications.

Keywords: Data Quality, Data Utility, quality of service, Cloud Computing, Fog Computing, data lifecycle, Data as a Service

1.1 Introduction

Due to the huge amount of data that are continuously produced, more and more data intensive applications are required to properly manage these data. Indeed, data need to be stored, retrieved, processed, and provided in an efficient way. This means that the data intensive applications need to be scalable, reliable, widely

accessible, also considering possible privacy restrictions.
Currently, several tools, focusing on different aspects have been proposed. For instance, HDFS [1] to provide a distributed file system, NoSQL databases [2] (e.g., MongoDB [3], Cassandra [4]) to ensure more flexibility and reliability, new programming models (e.g., MapReduce [5]), as well as new architectures (e.g., Lambda Architecture [6]) to improve the scalability.
It is worth noticing that most of the approaches proposed so far goes in the direction of making the data processing more efficient. For this reason, most of the solutions are conceived to run on cloud resources as they provide an environment ensuring scalability, reliability, and security.
Nevertheless, if we consider not only the data processing but the entire data management life-cycle, it is clear how cloud cannot be considered the only option. In fact, especially in IoT scenarios, data are produced on the edge (e.g., machineries, deployed sensors), then moved to the cloud where they are stored and processed and, finally, sent back to the same place where they are generated to, for instance, better configure the machinery that has been monitored, or to create reports used by the operators of those machines. This simple example highlights how data continuously move from the edge to the cloud and vice-versa and, due to the network infrastructure, a significant latency could be introduced.
A reaction to this situation has been to leave the data processing to where the data are produced. This is the Edge Computing [7] and it works perfectly when the data processing does not require a significant amount of resources, as they need to leave only on the edge of the network.
At the same time, Fog Computing has been arising as a platform able "*to provide compute, storage, and networking services between cloud data centers and devices at the edge of the network*" [8]. Similarly, as proposed by the OpenFog Architecture [9], we can consider Fog Computing as the sum of Cloud and Edge Computing where these two paradigms seamlessly interoperate [10].
In such a scenario, the approach proposed in this chapter relies on the Service Oriented Computing paradigm where data sets are exposed as Data as a Service (DaaS). This requires a proper description of the data to the final user where, in addition to the functional aspects, it includes the non-functional aspects concerning the location in which the data are stored, the format used, and the quality of such data. Since the user will base the decision of using or not this data source according to her/his needs, the more dimensions we put in the description the more the variable the user has to consider, thus the less usable will be the approach. For this reason, we introduce the notion of Data Utility, defined as the relevance of data for the usage context, where the context is defined in terms of the designer's goals and system characteristics. The developers' requirements are expressed in terms of functional (e.g., the application need to extract the data about the exams of the patients performed in the last year) and non-functional (e.g., latency, data completeness) aspects as well as constraints (e.g., data must be stored encrypted). The model of Data Utility has been discussed in [11].
If, on the one hand, users can take advantage of this concise description, on the other hand a higher burden is left to the providers. Indeed, the heterogeneity of the devices living on the edge or on the cloud, the influence of the network, the inherent Data Utility of the data sources are dependent to each other, and the goal

of this chapter is to model such dependencies.
The chapter is organized as follows. Sect. 1.2 gives an overview of the state of the art in this field. To better clarify the need and the usage of the Data Utility model, we use the running example introduced in Sect. 1.3. Sect. 1.4 discusses the conceptual model of data intensive applications running in a Fog environment. Sect. 1.5 introduces Data Utility and the concepts related to Data Utility definition and evaluation. Sect. 1.6 presents the lifecycle of the data sources and their Data Utility from the application developer and the data owner perspectives. Sect. 1.7 discusses the implication of Data Utility during the application deployment. Sect. 1.8 concludes the chapter outlining possible future directions.

1.2 Related Work

The concept of Data Utility has been used in several contexts. Various definitions have been provided in the literature. For the general IT context, Data Utility has been defined by [12] as "*The relevance of a piece of information to the context it refers to and how much it differs from other similar pieces of information and contributes to reduce uncertainty*". The value of data has been also of interest in the business scenario, where Data Utility has been defined as "*business value attributed to data within specific usage contexts*" [13]. A more complex definition has been given by [14] for Data Utility in the statistics context: "*A summary term describing the value of a given data release as an analytical resource. This comprises the data's analytical completeness and its analytical validity*".
The common factor between all these definitions is that the utility of a data set cannot be considered independently from the context in which data are used. Defining the context is not trivial and several elements can be included in its definition. Moreover, the concept of context can change dynamically, including new elements that were not relevant before. For instance, the concept of Data Utility has been applied in its early stage in the information economics area. As discussed in [15], the economic factors which are considered relevant for the assessment of Data Utility are the predicted benefits and costs of: (i) the analysis of the dataset, (ii) using the results of the analysis, and (iii) building the algorithms for executing the analysis of the dataset.
An important characteristic of the Data Utility concerns its variation with respect to the specific goal of the data analysis. As an example, in [16] Data Utility is analysed for data mining applications, while in [17] the focus is on users' requirements. Moreover, Data Utility might be influenced by the quality of service and the quality of data. For instance, the relation between Data Utility and quality of service has been investigated in [18], which discusses Data Utility with a focus on energy efficiency of mobile devices in a mobile cloud oriented environment. The issue of energy efficiency for discovering interrelations between the evaluation of the data value and the effectiveness of run-time adaptation strategies has been discussed in [19]. Similarly, the influence of data quality on Data Utility

is considered in [20], where relevant quality dimensions (e.g., accuracy and completeness) are considered in relations with Data Utility.
Finally, Data Utility has been also analysed with respect to the relation between IT and business, and this has paved the way for associating Data Utility to business processes. In this context, Data Utility is defined as a measurement of the gain obtained by using a dataset inside an organization [21]. Moreover, [22] discusses which are the information quality requirements in order to obtain reliable results from the execution of business processes.
The aim of this chapter starts from the results available in the state of the art for providing a definition of Data Utility comprehensive of all the relevant aspects emerged by this analysis: the context in which data are used, the content of the data, and the execution environment. We also evaluate how different configurations in terms of resource and data placement can affect Data Utility by using the concept of data movement introduced in [23] as a strategy for improving Data Utility.

1.3 Running Example

Being Data Utility a context-dependent aspect, a proper discussion about it requires the definition of a running example. In particular, this chapter relies on a scenario related to smart buildings. In more detail, we consider a building in which each room is monitored with sensors which allow a computing infrastructure to collect information about humidity, brightness and temperature. Sensors store this information at the edge of the network and make it available to several applications which want to manage the smart building. Additional data sources can be provided as DaaS and integrated with the information collected by the sensors. As an example, weather data provided by external services can be used to support some analyses.
In this running example, we focus on a specific data intensive application which uses as input all the illustrated data sources. The goal of the application is to analyse and improve the comfort in the building. To reach this goal, the application executes several tasks:

- *Task A - Ambient Sensing Alignment*: this task collects data coming from the temperature, humidity, and brightness sensors located in the rooms of the monitored building. These data are pre-processed to prepare them for further analysis (e.g., timestamps alignment and data cleaning).
- *Task B - Ambient Sensing Aggregation*: the pre-processed data obtained in the previous step are analysed to obtain some statistical information like maximum, minimum, and average value for each sensor. Also, data of similar sensors are compared to obtain a typical behaviour for each kind of sensor (e.g., average temperature in the rooms of the building). This statistical information is stored to be used in further steps of the application.
- *Task C - Data Enrichment and Prediction*: this task performs predictions about the status of the rooms in the building by using the data generated

in the previous step integrated with external information about the weather of the zone or city in which the building is located.

- *Task D - Visualization Preparation*: this task is in charge of presenting the results of the analysis performed in the previous tasks to the final user through visualization tools.

The structure of the application and the data sources involved in the analyses performed by its tasks are shown in Fig. 1. Tasks are represented as boxes labelled with the name of the task. The order in which the tasks are executed is modelled as a flow, represented through arrows connecting the different tasks. Data sources (DS) are modelled as inputs for the tasks. Two data sources are depicted: sensors of the building (labelled as DS_B, where B stands for Building sensors) representing streaming data collected from sensors, and databases containing weather data, labelled in the figure as DS_W, where W stands for Weather data. It is worth noticing that the output of a task can be considered as an input for the following task. This input is depicted in the flow between task t_i to task t_j as $E_{i,j}$, i.e. the information flow between *Task A* and *Task B* is labelled as $E_{A,B}$. In the figure, this exchange of information is depicted as a data object attached to the flow between two tasks with a dashed line and labelled as described.

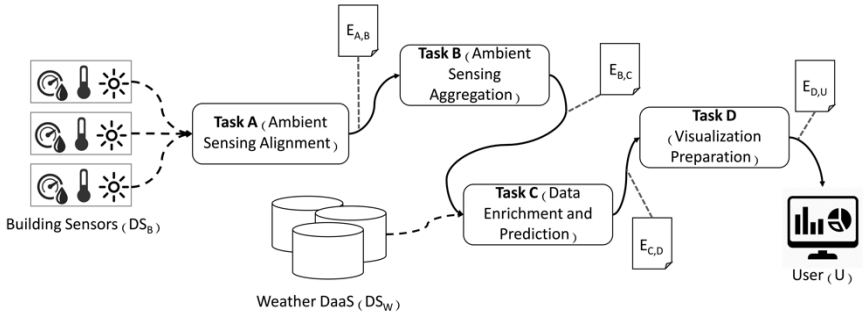

Fig. 1: Example of Data intensive application in ambient intelligence domain [11].

Other relevant information about the input of the tasks is related to the owner of the data source: data can be generated by sensors controlled by the application administrator (e.g., sensors in the building are the input of *Task A*, referred as DS_B) or by an external DaaS (e.g., weather data are the input of *Task B*, referred as DS_W). Moreover, the location of the data source is another element which affects the application (e.g., sensor data might be stored in an edge resource or in a cloud storage).

While for *Task A* only a specific data source can be selected, identified in the sensors of a specific building, in other cases alternative data sources might be associated with a task. This is the case of *Task C*, where the input is weather information which can be retrieved by different DaaS providers fitting the requirements of the application. In this context, selecting the data source that better fits the requirements might be not trivial. In fact, several variables influence the efficiency and effectiveness of the task. One variable is related to the relative location of the task and the data source. As an example, placing *Task A* at the edge of the network might improve the execution of the task reducing the amount of

data collected by the sensors in the building that need to travel from the edge to the cloud. Similarly, also *Task B* might be conveniently located in the edge. Another variable to consider is the type of resources available in the edge and in the cloud. In fact, computation in the edge is usually executed on devices with limited capabilities and performance, and this might have a negative impact on the application, creating a bottleneck. Finally, in case of alternative data sources, we might take into consideration that different data sources provide data with different quality of data (e.g., the different weather data sources could have different quality levels). The selection of a provider instead of another one eventually affects the quality of the analysis performed by the whole application.
All the described variables might be considered during the deployment of the application, making the work of the application developer really hard. To support him/her in this job, Data Utility provides a complex metric reflecting to which extent a data source satisfies the requirements of an application. The introduction of Data Utility can reduce the effort of the application developer in the selection of the data sources and the tasks location.

1.4 A model for data intensive applications in Fog environment

Data intensive applications are applications whose main goal is to manage a significant amount of information. They can be therefore defined by the data that they receive as an input, the processing steps executed to get results from them, and where the data are initially stored as well as they are stored during the processing steps. According to this, data intensive applications are usually modelled as a data flow between several processing steps. In this context, it is important to capture the relation existing between the data and the tasks operating on them. In [24], this relation is represented through a UML profile. A more refined attempt of modelling data intensive applications is presented in [25], where they are modelled from different perspectives (from business to technical aspects).
With respect to the existing work, a peculiar aspect of the approach discussed in this chapter relies on the adoption of the Fog Computing paradigm for designing and running the application. As a consequence, the model adopted to define a data intensive application mainly goes in the direction of specifying how Fog Computing may affect the design and the execution of this type of applications.
Fog Computing gets advantage of the capabilities of cloud extending them towards the edge of the network. In this way, the computation and data storage can be moved nearer to the final user, improving response time and reducing latency. The tools supporting Fog Computing hide the complexity and heterogeneity of the environment and data intensive applications can consider cloud and edge devices as a continuum. From a designer standpoint, instead of specifying a precise deployment plan, it is more interesting to specify which are the characteristics that a node should, or must, have to run a task or to store data. At design time,

movement of data and tasks between devices in the cloud or in the edge can be enacted to improve the efficiency and effectiveness of applications during their execution.

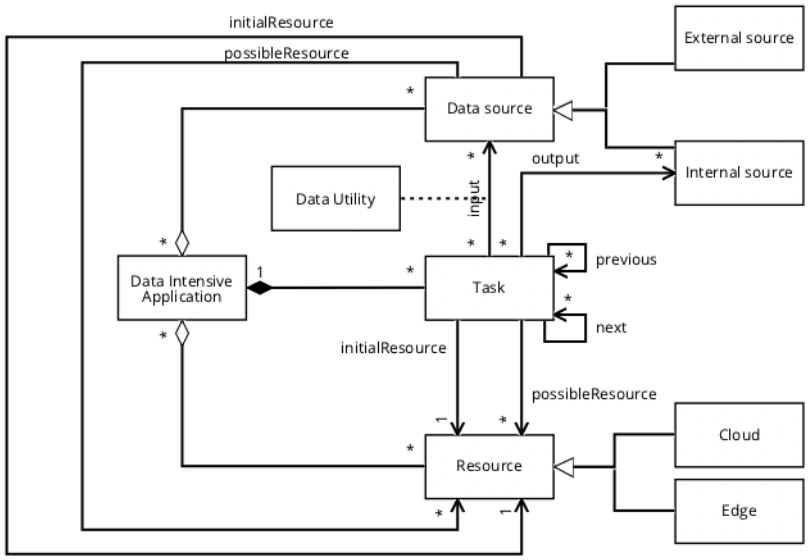

Fig. 2: Data intensive application model [11].

Based on these assumptions, and focusing only on the design standpoint that reflects the main goal of this chapter, we can model a data intensive application as composed of three main elements: *resources*, *data sources*, and *tasks*. Fig. 2 reports some of the characteristics of these elements with an emphasis on the relationships among them using the UML notation for class diagrams. In the model, a data intensive application is represented as a composition of tasks and its related to resources and data sources. All these three main elements are described in details in the following paragraphs.

1.4.1 Resource

Resources are the nodes in which the application can be deployed and in which data can be stored. According to the Fog paradigm, resources can be placed either in the cloud or in the edge on heterogeneous devices, including Virtual Machines, sensors, laptops, and smart devices. The computational and storage capabilities of these nodes are of interest since they might affect the execution of the application and its performance. Knowing this information about a resource makes it possible to understand if it is suitable for hosting the application or the data consumed

during its execution. According to this, when describing a resource, we need to define the computational capabilities in terms of number and frequency of CPUs, and the storage capabilities in terms of both RAM and disk capacity. The exhaustive description of the hardware features of a node is not of interest in this chapter, but standard approaches as the DMTF-CIM [26] can be adopted.
In Fig. 2 resources are classified in cloud and edge using the generalization notation of UML. In fact, this distinction is not only due to the location of the resource, but also to its features. First of all, one difference is related to the ownership of the resource. Usually, a cloud infrastructure is not managed by the application developer but belongs to an external entity which provides only a limited set of information about, for instance, the location and features of the resource itself. For instance, when deploying a VM on a cloud infrastructure, the cloud provider usually shares the information on the site in which the VM has been deployed but it is not possible to access any information on the specific physical machine hosting it. Yet, the application developer has no control on migration of the VM from one server to another since this decision is managed by the cloud provider. In the case of an edge resource, we assume that the application developer can use only resources that s/he owns. According to this, the developer has a full knowledge and control on these resources, and can reconfigure and adapt them during the execution. As an example, we assume that it is possible to adjust the sampling rate of a sensor for making it in line with the needs of the application.

1.4.2 Data Source

The second element of data intensive applications that we want to analyse is the data source. The data source indicates the input of the application, including all the data required for executing the analysis and the relevant information written during the execution. Adopting the Service Oriented Computing paradigm, data sources are exposed as DaaS, thus we model them accordingly.
From a functional perspective, APIs can be used to define which are the type of data that the data source makes available [27], as well as information is related to how data can be accessed (i.e., endpoints). Being independent from a specific implementation, with APIs it is possible to abstract from specific resources, and this is an advantage when dealing with heterogeneous environments as in the case of Fog Computing. In this way, the movement of a data source does not affect how the tasks access the content of the data source. Due to the different nature of data sources available, also different interactions are possible. We distinguish between conventional interactions, where data are accessed through queries, and stream interactions, which better reflects sources as sensors and monitoring systems, where data are continuously produced and updated.
A data source is stored in one of the resources described in the previous paragraph (either on the edge or the cloud), but it can be moved in other resources taking advantage of data movement in Fog environments. Following the approach in [24] data sources are classified in *Internal Sources* containing data that can be managed

by the application designer, and *External Sources* which are data provided by an external service. The former class includes also the data produced by the application during its execution and exchanged between its tasks, as well as data stored in edge and cloud devices but owned and/or managed by the application developer. In Fig. 1, examples of this kind of sources are the data produced by sensors (DS_B) and data exchanged among tasks ($E_{i,j}$). External sources are managed externally from the application, as the weather data DS_W in the example.
Finally, a data source is associated with the resources. With *initial resource,* we indicate the resource in which a data source is deployed at the moment of the Data Utility evaluation. With *possible resources,* we indicate the set of storage nodes satisfying functional constraints of the data source (e.g., storage size and file system) in which the data source can be moved.
A data source described in this way can be associated with a Data Utility value expressing its relevance in a given context. Using the model for data intensive applications introduced in this paragraph, Data Utility characterizes the association between a task, using a data source as an input, and a data source providing the data. Extending classical data quality evaluation, Data Utility is a multidimensional evaluation including attributes like accuracy, reliability, timeliness, precision, completeness [28]. Data Utility is affected by the resources involved in the deployment for both hosting the tasks and storing the data sources, and their location.

1.4.3 Task

A data intensive application can be seen as a sequence of tasks, each one with inputs and outputs, each of them cooperating in the analysis of the data set for extracting the desired knowledge. As discussed in [25], a task is a unit of work with a duration in time and, in the context of data intensive applications, we can identify a specific set of tasks: e.g., data cleaning, data integration, compression, and encryption. Typical tasks are associated with typical algorithms, but can be personalized and integrated with custom scripts. A data flow model can be used to express how data are acquired, transformed, and returned by the tasks composing the data intensive application [29]. To represent the order in which tasks are executed, the flow of tasks is expressed using the *next* and *previous* attributes. Tasks are also associated with the data sources which contain their *inputs* and host their *outputs*. An output is always connected to a data source which is an internal source, since the data produced by the task are managed internally. On the contrary, an input can be stored both in an internal source (in case of data exchanged between tasks or data source managed by the application developer) or an external source (in case of a data source managed externally from the application).
Tasks are deployed on computational resources available in the described Fog environment. Similarly, to the data sources, a task is associated with the resource in which it is initially deployed (*initialResource*) and with a list of computational resources which are able to host the task (*possibleResources*).

1.5 Data Utility in Fog Environments

As discussed in Sect. 1.2, Data Utility has been defined in different ways in the literature. All these definitions agree on its dependency on the context in which data are used. Therefore, the assessment of Data Utility is a complex issue since context can be composed of several elements and it usually changes over time. Moreover, Data Utility aims to provide an indication of the relevance of data for the usage context that is defined in terms of system characteristics and application designer's goals. Therefore, before to discuss Data Utility in details we present in the following the components of the usage context.

1.5.1 Usage context

As stated above, the usage context depends on the status of the system and the application designer requirements. According to the proposed model introduced in Sect. 1.4, Data Utility is defined as an association class among tasks and data sources.
Starting from the available data sources, for each data source, it is important to know the data source schema (S_j), the initial resource (ir_j) on which it is currently deployed and the set of possible resources on which it could be deployed (PR_j). Thus, the set of all the available data sources *DS* can be represented as:

$$DS = \{ds_j\} = \{< S_j, ir_j, PR_j >\}$$

where ds_j is the single data source and each data source is defined by a data source schema S_j, an initial resource ir_j, and a set of potential resources PR_j.
Moving to the tasks that compose the data intensive application, each task should be described by the application developer in terms of:

- a description of the operations performed by the task D_i (e.g., aggregation, filtering, clustering, association);
- the set of inputs and outputs IN_i and OUT_i;
- the position of the task within the data flow in terms of the set of tasks that precede and follow (P_i and N_i);
- the current resource ir_ion which the task is deployed and the resources on which it can be potentially deployed PR_j the analysed task in the data-flow process.

Formally, we assume that each task t_i is defined by:

$$t_i = < D_i, IN_i, OUT_i, P_i, N_i, ir_i, PR_i >$$

It is worth noticing that, as also depicted in Sect. 1.3, tasks may gather inputs:

1. from a specific data source (i.e., *Task A*);
2. from a previous task (i.e., *Task B*);
3. from a data source that should be selected from a set of candidate sources (i.e., *Task C*).

We can see the first and the second case as situations in which the task relies on a single and known source. Conversely, the last case concerns the situation in which the application developer could need support in the selection of the sources.
To better distinguish among these situations and to also consider that a task may have several inputs, we can model the *k*-th input of the *i*-th task (IN_{ik}) as:

$$IN_{ik} = < A_{ik}, CDS_{ik} >$$

where, A_{ik} is the set of the attributes of the data source required by the task (e.g., temperature, humidity), and CDS_{ik} the set of candidate data sources from which data have to be extracted. Since candidate data sources are a subset of the data sources available according to the needs of the designer, we can say that $CDS_{ik} \subseteq DS$. Such set can include both internal and external sources. If the cardinality of this set is more than one, it means that the designer would like to be supported in the identification of the most suitable source among the ones available. In this case, for triggering the selection procedure a Data Request R_{ik} is created for capturing the application designer's goals, i.e., the elements that can affect the source selection. The Data Request R_{ik} is composed of three elements:

$$R_{ik} = < IN_{ik}, {f_{ik}}^{*}, {NF_{ik}}^{*} >$$

where:

- IN_{ik} is the *input definition*: a task may require several inputs. For each input the application designer has to specify the list of attributes that the task needs and the set of candidate data sources from which data have to be extracted;
- ${f_{ik}}^{*}$ is the list of *functional requirements*: the * indicates that it is an optional parameter used to express functional requirements. It can be seen as a predicate, composed of atoms linked by traditional logical operators (i.e., AND, OR), that allows the designers to specify restrictions over the allowed values in order to better drive the source selection (e.g., city= "Milan" AND Temp > 23);
- ${NF_{ik}}^{*}$ is the list of *non-functional requirements*: also optional, they contain requests related to Data Quality, QoS, and security aspects. Note that all the non-functional requirements can be expressed by the developers or automatically gathered considering the kind of task. Quality requirements include:
    - Data quality requirements: they focus on the quality of the content provided by the source;
    - QoS requirements: they focus on non-functional properties such as availability, latency or cost. Since the assessment of such

properties mainly depends on the resources on which the task or the data are deployed/stored, the evaluation whether QoS requirements are met has to consider the placement of the tasks and data sources.

Each non-functional requirement is an expression with which the related constraint to a DQ or QoS dimension is specified (e.g., data source completeness > 0.9 or latency < 10 sec).

Therefore, for example, if an application wants to analyse the data related to the temperature values collected in a specific room of the building in the month of August, the data request can be the following:

- *Date, time, temperature*
- *Date BETWEEN 01/08/2017 AND 31/08/2017*
- *latency < 60 sec AND accuracy >99% AND completeness >98%*

It is possible to enrich the request with additional non-functional constraints that can be derived by the type of task. For example, a task that performs data mining operations requires a high amount of data and high completeness. If the developer has not specified requirements on such dimensions, the system, by analysing the type of task, will add these constraints to improve the effectiveness of the source selection.

1.5.2 Data Utility

The fitness of a data source to the application developer's requirements is expressed through the evaluation of the Data Utility. This evaluation is led by the request R_{ik} provided by the users together with the characteristics of the data source and of the task that requests it. For each data source provided by a DaaS, the Data Utility evaluation is performed to assess which is, among the alternative sources, the one better fitting the application developer's requirements. In particular, Data Utility evaluation has to take into account:
the capability of the source to satisfy the functional requirements of the task: the degree with which the source contains the data requested by the application;

- the capability of the source to satisfy non-functional requirements of the task: the degree with which the source is able to satisfy quality requirements;
- the reputation of the source: the degree with which the source is trustworthy.

More formally, we assume that the data sources are associated with a set of metadata that reveal the *Potential Data Utility* (PDU) that summarizes the capabilities of the data source and can be periodically evaluated independently of the context. The PDU is calculated looking at the data and the characteristics of the data source. It is derived from a *Data Quality* and a *Reputation* assessment. From a Data Quality perspective, it is important to highlight that errors, missing

data, or updated data might significantly affect the usage and potential benefits of data. The assessment of Data Quality dimensions may contribute to the understanding of the potential value of the data. The list of dimensions (e.g., accuracy, consistency, completeness, timeliness) and the assessment metrics depend on the type of data contained in the source. For example, in case of data collected from sensors precision and data stability constitute the two most relevant dimensions to take care of. Generally speaking, we assume that each source is associated with a set of Data Quality dimensions and related values. Besides the content, also the usage of the source is considered and defined as Reputation index. This index depends on the frequency with which the source has been successfully used and on the scope of data (e.g., generic or specific, be integrated with other sources, used with other sources).

A data source has to be evaluated by considering the context that in our scenario is composed of a data intensive application and the available resources. QoS capabilities have to be evaluated by considering all the available options that the Fog environment offers. Thus, both tasks and data can be moved: (i) from edge to cloud, (ii) from cloud to edge, (iii) from edge to edge, and finally (iv) from cloud to cloud. Variation of the placement of a task or a data source on a specific resource has surely an impact on the QoS: in fact, the computational cost for obtaining data and the latency changes on the basis of the chosen location. For instance, it is reasonable to assume that the ambient sensing alignment task can be more efficient if it is executed closer to the sensor data to be aligned. As we assume that both data and task can be moved, we calculate the QoS dimensions for each possible configuration defined in terms of task placement associating a task t_i to the resource r_x in which it is deployed ($< t_i, r_x >$) and data placement associating a data source ds_j to the resource r_y where it is stored ($< ds_j, r_y >$).

Data Utility of a data source ds_j for a task t_i is defined as:

$$DU_{ixjy} = f(< t_i, r_x >, < ds_j, r_y >)$$

Both tasks and data sources, according to our data intensive application model, can be placed on different resources belonging to PR_i and PR_j. Data Utility depends also on the task placement ($< t_i, r_x >$) and data sources placement ($< ds_j, r_y >$) where $r_x \in PR_i$ and $r_y \in PR_j$.

In summary, as shown in Fig. 3, Data Utility can be assessed by considering three main aspects: Data Quality, Reputation, and Quality of Service. Each of them is evaluated by means of *Dimensions*, each one associated with different *Metrics* (more than one assessment function might be available for a single dimension).

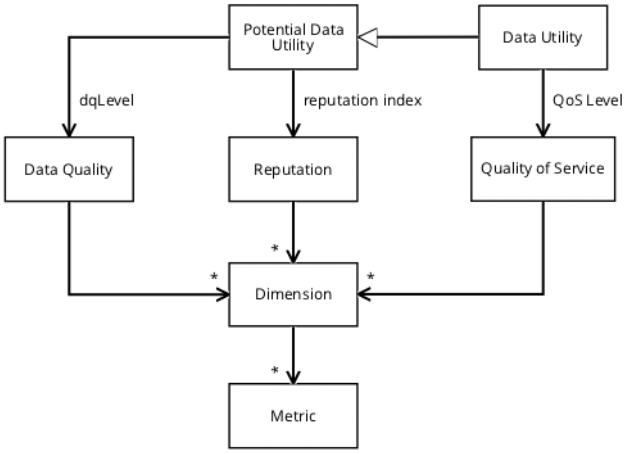

Fig. 3: Model of the utility components [11].

1.6 Data Lifecycle

As Data Utility evaluation is dynamic and change over time according to the variation in terms of its components (e.g., data quality, quality of service, and reputation), we can describe the lifecycle of a data source from two perspectives: the DaaS perspective and the application developer's perspective. The former considers the data source in the traditional lifecycle phases: from the data source creation to the disposal. The latter considers the data source as an object to access and manipulate.

Focusing on the DaaS perspective, the data lifecycle is represented in Fig. 4. Firstly, the data source, in order to be managed by the platform, has to be registered, and the registration requires the definition of metadata that describe the source and its contents. Once that the source is registered, the Potential Data Utility can be evaluated as all the elements independent from specific context in which data are used (e.g., accuracy, completeness, consistency) can be obtained. In this way, the source is enriched with PDU metadata that are one of the main drivers for the selection of data with respect the applications requests. After this phase, the source is available for use. Periodically, the PDU is re-evaluated and corresponding metadata are updated. All the criteria of the Data Utility are instead assessed when an application sends a request. Data Utility evaluation considers the different variants of the context on the basis of the location of the application and the data source. This means that, for each request, several Data Utility vectors are calculated and a ranking can be defined.

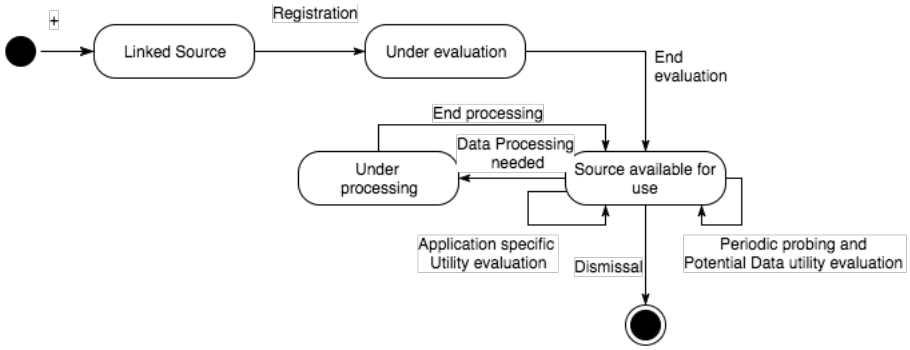

Fig. 4: Data Lifecycle from the DaaS perspective.

Moving to the application developer's perspective, the data lifecycle starts with the submission of the data request in which the application developer specifies the data sources, and the functional and non-functional requirements (see Fig. 5). Once the set of valid data sources is identified on the basis of the functional requirements, each source is enriched with the utility scores that are evaluated considering the application and data source status. On the basis of these data, the application developer makes the final decision and the selected data source is instantiated and bind to the application.

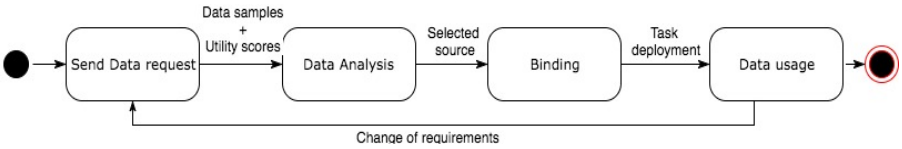

Fig. 5: Data lifecycle from an application perspective.

1.7 Using the Data Utility Model

Given a task and a data source composing the data intensive application, the evaluation of the Data Utility considers all the possible configurations, in the edge or in the cloud, of both the data source and the task. At this stage, the Data Utility model returns an evaluation based on three main dimensions, i.e., Reputation, Data Quality, and QoS. Thus, a tool supporting the designer by adopting the proposed Data Utility model can show, for each task and each configuration, where more than a data source is available (e.g., *Task C*), and how these three dimensions vary (see Fig. 6)In case a ranking of the different alternatives is required, different methods for identifying the best data source are available. They range from a simple aggregation (e.g., average or weighted average) of the

different dimensions to more advanced techniques for multiple-criteria decision analysis (i.e., MCDA methods).

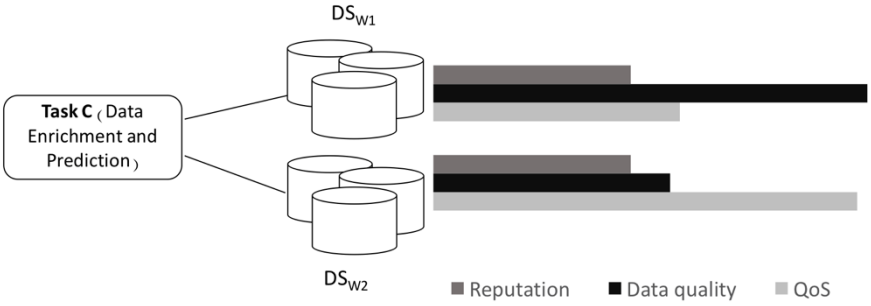

Fig. 6: Using Data Utility model to evaluate alternative data sources [11].

Although the proposed model provides a significant and useful tool for data intensive application designers to understand the impact of moving data and tasks with respect to the Data Utility, this evaluation focuses only on a task level analysis. For this reason, we advocate the need of a global Data Utility model that is able to capture the Data Utility of the entire application based on the selection of the different sources. At this stage, the definition of a Global Data Utility measure, that is the utility of the results provided to the final user, is under investigation, and here we would like to outline which are the main elements to be considered.

First of all, we want to highlight that an increasing number of tasks and possible data sources imply an exponential increasing of the Data Utility evaluations required. This increment can be mitigated by the number of constraints that the designer can put to the requests in terms of allowed data movements. For instance, *Task A* and source DS_B are only considered in the edge because of a constraint forbidding to move the data produced by the sensors. For simplification, we assume that each task can be moved in the edge only if its predecessor is already in the edge (otherwise moving it does not provide any advantage since communication is not improved). This is true for our example, but not generally true when several tasks use data produced by edge devices.

To give an idea, referring to the example discussed in Sect. 1.3, the different possible configurations are summarized in Table 1, where only two Weather Data providers are considered: DS_{W1} and DS_{W2}. The table is horizontally divided in four sections, each one representing a possible task deployment configuration, where on the right all the combinations about data source selection and placement is considered.

Table 1: Deployment alternatives in a Fog Environment.

	Task A	*Task B*	*Task C*	*Task D*	DS_B	DS_{W1}	DS_{W2}
Depl. 1	Edge	Edge	Edge	Edge	Edge	Edge	--
Depl. 2	Edge	Edge	Edge	Edge	Edge	Cloud	--
Depl. 3	Edge	Edge	Edge	Edge	Edge	--	Edge
Depl. 4	Edge	Edge	Edge	Edge	Edge	--	Cloud
Depl. 5	Edge	Edge	Edge	Cloud	Edge	Edge	--
Depl. 6	Edge	Edge	Edge	Cloud	Edge	Cloud	--
Depl. 7	Edge	Edge	Edge	Cloud	Edge	--	Edge
Depl. 8	Edge	Edge	Edge	Cloud	Edge	--	Cloud
Depl. 9	Edge	Edge	Cloud	Cloud	Edge	Edge	--
Depl. 10	Edge	Edge	Cloud	Cloud	Edge	Cloud	--
Depl. 11	Edge	Edge	Cloud	Cloud	Edge	--	Edge
Depl. 12	Edge	Edge	Cloud	Cloud	Edge	--	Cloud
Depl. 13	Edge	Cloud	Cloud	Cloud	Edge	Edge	--
Depl. 14	Edge	Cloud	Cloud	Cloud	Edge	Cloud	--
Depl. 15	Edge	Cloud	Cloud	Cloud	Edge	--	Edge
Depl. 16	Edge	Cloud	Cloud	Cloud	Edge	--	Cloud

A second aspect to be considered concerns the mutual influences between tasks, making the data source selection and movement a complex decision. Two factors should be considered in the global Data Utility computation:

- *The Data Utility of a task* t_i *influences the Data Utility of a task* t_j *with* t_i *successor of* t_j. It means that the selection of a data source for task t_i has an effect on t_j which uses the output provided by t_i, both directly and indirectly. In the example depicted in Fig. 1, selecting a data source DS_{Wx} optimizing the Data Utility for *Task C* impacts on the Data Utility of *Task B* in which different requirements could make a data source DS_{Wy} more convenient for the global Data Utility. On the computation movement perspective, this dependency is also relevant. As an example, moving *Task B* to the edge may improve its the Data Utility by allowing a faster access to its input (generated by *Task A*), but affects the

performance of *Task C* for which a higher latency in data retrieval must be considered.

- *The Data Utility of a task t_j influences the Data Utility of a task t_i with t_i predecessor of t_j*. Maximizing the Data Utility means selecting the best coupling between a task and a data source according to the task requirements, while maximizing each components of the Data Utility model. As an example, focusing on source selection, if both DS_{Wx} and DS_{Wy} satisfy the task requirements, DS_{Wx} could be selected because of a better timeliness if compared with DS_{Wy}. However, t_j requirements could be violated by DS_{Wx}, and in this case DS_{Wy} would be preferable.

Finally, providing techniques and heuristics for selecting the deployment configuration maximizing the global Data Utility for the user of the application is out of scope, but is an interesting challenge that might be considered in the future.

1.8 Concluding Remarks

Nowadays, the amount of available data sources is continuously increasing. This is mainly due to the fact that new technologies and applications allow us to transform many aspects of our life into digital data. Data intensive applications analyse such data in order to understand them and support decision making processes, the design of advanced services, or the optimization of existing processes.

However, having a high quantity of accessible data sources is not always an advantage for the application developers. In fact, they may experience some difficulties in selecting the appropriate source for their goals. For this reason, in this chapter we have introduced the Data Utility concept that is able to evaluate the relevance of a data source along the usage context. The context is defined in terms of both the developers' requirements and the status of the system in which data sources and applications are stored and deployed. In particular, we define our approach by considering data intensive applications running on a Fog environment. In such scenario, the location of both tasks and data sources can be changed by using cloud or edge resources; therefore, the influence of data and computation movement is also considered in the Data Utility definition. Currently, the presented contribution supports the Data Utility driven selection of the data sources at the task level. However, the evaluation of the optimal utility for an application as a whole is under investigation to consider the influences among tasks, as well as the constraints over the deployment.

Acknowledgements

This research has been developed in the framework of the DITAS project. DITAS project receives funding from the European Union's Horizon 2020 research and innovation programme under grant agreement RIA 731945.

References

1. Borthakur D. (2008) HDFS architecture guide, Hadoop Apache Project
2. Pokorny J. (2013) NoSQL databases: a step to database scalability in web environment. International Journal of Web Information Systems. Mar 29;9(1):69-82
3. Chodorow, K. (2013) MongoDB: The Definitive Guide: Powerful and Scalable Data Storage. O'Reilly Media, Inc.
4. Apache Cassandra, http://cassandra.apache.org (last access 26/01/2018)
5. Bhandarkar M. (2010) MapReduce programming with apache Hadoop. In: Parallel & Distributed Processing (IPDPS), 2010 IEEE International Symposium on 2010 Apr 19 (pp. 1-1). IEEE.
6. AWS Lambda, https://aws.amazon.com/lambda/ (last access 26/01/2018)
7. Shi, W., Dustdar, S. (2016) The Promise of Edge Computing. Computer 49(5), 78–81
8. Bonomi, F., Milito, R., Zhu, J., Addepalli, S., Fog computing and its role in the internet of things, In Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, pp. 13–16. MCC '12 (2012)
9. OpenFog Consortium Architecture Working Group, OpenFog Architecture Overview (February 2016), http://www.openfogconsortium.org/ra
10. Plebani, P., Garcia-Perez, D., Anderson, M., Bermbach, D., Cappiello, C., Kat, R.I., Pallas, F., Pernici, B., Tai, S., Vitali, M., Information Logistics and Fog Computing: The DITAS Approach, In Proceedings of the Forum and Doctoral Consortium Papers Presented at the 29th International Conference on Advanced Information Systems Engineering, CAISE 2017, Essen, Germany. pp. 129–136. CEUR Vol-1848 (2017)
11. Cappiello, C., Pernici, B., Plebani, P., Vitali, M., Utility-Driven Data Management for Data-Intensive Applications in Fog Environments, In International Conference on Conceptual Modeling (pp. 216-226), Springer, Cham ((2017)
12. Kock, N., Encyclopedia of E-collaboration. Information Science Reference, Imprint of IGI Publishing, Hershey, PA (2007)
13. Syed, M.R., Syed, S.N, Handbook of Research on Modern Systems Analysis and Design Technologies and Applications, Information Science Reference, Imprint of IGI Publishing, Hershey, PA (2008)
14. Hundepool, A., Domingo-Ferrer, J., Franconi, L., Giessing, S., Nordholt, E.S., Spicer, K., de Wolf, P.P., Statistical Disclosure Control, John Wiley & Sons (2012)
15. Weiss, G.M., Zadrozny, B., Saar-Tsechansky, M., Guest editorial: Special issue on utility-based data mining, Data Mining Knowledge Discovery 17(2), 129–135 (Oct 2008)
16. Lin, Y.C., Wu, C.W., Tseng, V.S., Mining High Utility Itemsets in Big Data, pp. 649–661. Springer International Publishing, Cham (2015)
17. Ives, B., Olson, M.H., Baroudi, J.J., The measurement of user information satisfaction, Commun. ACM 26(10), 785–793 (Oct 1983)
18. Wang, R.Y., Strong, D.M., Beyond Accuracy: What Data Quality Means to Data Consumers, Journal of Management Information Systems 12(4), 5–33 (1996)
19. Ho, T.T.N., Pernici, B., A Data-Value-Driven Adaptation Framework for Energy Efficiency for Data Intensive Applications in Clouds, In Technologies for Sustainability (SusTech), 2015 IEEE Conference on. pp. 47–52. IEEE (2015)
20. Moody, D., Walsh, P., Measuring the Value of Information: An Asset Valuation Approach, In European Conference on Information Systems (1999)
21. Even, A., Shankaranarayanan, G., Berger, P.D., Inequality in the utility of customer data: Implications for data management and usage, Journal of Database Marketing & Customer Strategy Management 17(1), 19–35 (2010)
22. Gharib, M., Giorgini, P., Mylopoulos, J., Analysis of information quality requirements in business processes, revisited, Requirements Engineering pp. 1–23 (2016)

23. D'Andria, F., Field, D., Kopaneli, A., Kousiouris, G., Garcia-Perez, D., Pernici, B., Plebani, P., *Data Movement in the Internet of Things Domain*, In Proc. Eur. Conf. Service Oriented and Cloud Computing, ESOCC 2015. pp. 243–252 (2015)
24. Gomez, A., Merseguer, J., Di Nitto, E., Tamburri, D.A., *Towards a UML Profile for Data Intensive Applications*, In: Proc. Int.l Workshop on Quality-Aware DevOps. Saarbrücken, Germany. pp. 18–23 (2016)
25. Nalchigar, S., Yu, E., Ramani, R., *A Conceptual Modeling Framework for Business Analytic*s, pp. 35–49. Springer International Publishing, Cham (2016)
26. Distributed Management Task Force Inc., Common Information Model (DMTF-CIM), https://www.dmtf.org/standards/cim
27. Cleve, A., Brogneaux, A.F., Hainaut, J.L., *A Conceptual Approach to Database Applications Evolution*, Springer Berlin Heidelberg (2010)
28. Batini, C., Scannapieco, M., *Data and Information Quality-Dimensions, Principles and Techniques*, Data-Centric Systems and Applications, Springer (2016)
29. Garijo, D., Alper, P., Belhajjame, K., Corcho, O., Gil, Y., Goble, C.A., *Common Motifs in Scientific Workflows: An Empirical Analysis*, Future Generation Computing Systems 36, 338–351 (2014)

Index