QoS-based Trust Evaluation for Data Services as a Black Box

Senda Romdhani, Genoveva Vargas-Solar, Nadia Bennani, Chirine Ghedira

QoS-based Trust Evaluation for Data Services as a Black Box

Senda Romdhani
Univ. of Lyon, CNRS
Univ. of Lyon 3, LIRIS
Lyon, France
senda.romdhani@univ-lyon3.fr

Genoveva Vargas-Solar
CNRS
LIRIS
Lyon, France
genoveva.vargas-solar@liris.cnrs.fr

Nadia Bennani
Univ. of Lyon, CNRS
INSA-Lyon, LIRIS
Villeurbanne, France
nadia.bennani@insa-lyon.fr

Chirine Ghedira-Guegan
Univ. of Lyon, CNRS
iaelyon - Univ. of Lyon 3, LIRIS
Lyon, France
chirine.ghedira-guegan@univ-lyon3.fr

***Abstract*—Under the black-box model, data services do not export (meta)-data describing the conditions in which data are collected, in which they are deployed and processed, and the quality of the data they deliver. Thus, this model creates blind spots that prevent determining to which extent providers can be trusted to use their data services for building target applications. This paper proposes a QoS-based trust evaluation model for black box data services that combines QoS indicators, including service performance and data quality. The paper also introduces DETECT (Data sErvice as a black box Trust Evaluation arChitecTure) which validates our model. The experimental results demonstrate the feasibility and effectiveness of our solution.**

***Index Terms*—Trust, Data Service, QoS, Data Quality, Performance**

I. Introduction

The explosion of data collected, managed, and provided using online services leads to the logic of "Everything as a Service" (XaaS: X as a Service). Currently, the use of "Data as a Service" (i.e., data service) for accessing large volumes of data from different providers concerns mainly data consumers who see services as practical and easy-to-use components for accessing data. In this context, it is necessary to identify trustworthy data services that can provide data under specific conditions and fulfilling quality requirements.

A trustworthy service respects the terms and QoS as promised by its provider. In a data service's ecosystem consisting of services with different QoS and distinct data quality properties, trust is also related to the degree of data quality that can be ensured and essential for data integration, among others.

Service Level Agreements (SLAs) specify clauses about the promised QoS for given services and penalties if the service provider violates these QoS clauses. However, SLAs rarely include data quality aspects in their agreements. To ensure that QoS agreements are respected, service providers deploy solutions to monitor their services by observing and computing QoS measures (i.e. service performance). The monitoring is performed to make internal technical decisions related to resource allocation. However, for privacy and security reasons, service providers rarely share their monitoring measures. Thus, they do not provide detailed proof that the performance of the services adheres to the promised QoS in the SLA. In this sense, data services adopt a black-box model under which services do not provide details on their backend, including how data are collected, updated, and to which extent data are fresh. Thus, the consumer chooses the service blindly. Still, clients are looking for alternatives to evaluate the trust level of both services and data as criteria to select services for building information systems and retrieving data.

To illustrate how challenging is the data service selection in the black-box context, let us consider an e-health scenario[1].

Fever in chemotherapy-induced neutropenia (FN) is the most frequent, potentially lethal complication in patients with cancer [1]. Fever is particularly dangerous if the white blood count becomes low. During this time, the body's normal defenses against infections are down. Thus, cancer patients' body temperature should be monitored continuously to react promptly under a medical emergency when fever is detected.

Consider the case of Alice, a 50-year-old woman with breast cancer treated with chemotherapy. To avoid FN complications when she is at home, she uses medical devices provided by the hospital's emergency service to keep track continuously of her temperature among other physiological measures. Devices are programmed to collect data at different rates. According to a specific update frequency, measures are sent to the hospital, maintaining a database with Alice's data. Alice also uses a connected thermometer to measure her temperature frequently. The device sends the data to an e-health service installed in her smartphone. She also measures her temperature using a regular thermometer every x days or weeks, depending on her mood. She records these measures in a service installed on her smartphone. Alice gives access to her health data through the services deployed on her smartphone. Doctors should access

[1]This scenario was proposed in the context of the project SUMMIT.

her data at any moment. Data must be reliable to be used effectively to make accurate decisions, and it must be delivered promptly in case of an emergency. Each service ensures different QoS and data quality guarantees since data are produced and updated under different conditions (production rates, update frequencies that affect data freshness).

A medical network can exchange data through services that must be carefully selected according to their degree of trust. Deciding which service to use to retrieve Alice's physiological data in a given situation can be critical for doctors as services have different trust levels. In ideal conditions, services must be regularly tagged with a trust measure that can be used selection criterion, computed using service performance and data quality factors. Therefore, the challenge is to propose (1) a trust model that captures and combines service performance and data quality aspects and (2) a mechanism to collect information required to compute services' trust levels.

The main contribution of our work is a trust evaluation model for data services and the architecture DETECT that implements it using an e-health scenario. The experimental results show how the trust level of services with different QoS can be computed and used to rank them given services' lookup requests.

The remainder of this paper is organized as follows. Section II presents the related work. Section III describes our QoS-based trust evaluation model. Section IV describes the DETECT architecture. The experiments to prove our solution's feasibility is presented in section V. Section VI concludes the paper and discusses future work.

II. Related Work

Services trust evaluation has been addressed in service-based environments, including SOA, web services, cloud services, etc. Existing proposals focus on trust at the (i) service and (ii) data levels.

a) Trustworthy Services: Service trust solutions adopt different trust assessment solutions: fuzzy [3], probabilistic [4], machine learning [5], multi-criteria decision-making (MCDM) [6] and classical mathematical such as weighted-addition [7]. Service trust evaluation can be subjective [8], objective [7] and hybrid [9]. Subjective evaluation relies on user preferences and feedback about service usage. Users' feedback may be quantitative or qualitative, provided in textual form. Several factors may influence users' subjective feedback. For a given service, such feedback may vary significantly from one user to another, and there is no guarantee of users' objectivity. Objective evaluation involves measurable metrics associated with the service capacities, capabilities, and performance [2]. Performance metrics are measured by monitoring services' behavior over time and measuring to which extent the SLAs are fulfilled [18], [19].

b) Trustworthy Data: Data similarity and data provenance are two of the most common features for assessing data trust. For similarity-based solutions, reliable data represent an event with similar values [11]. For provenance-based solutions, data with reliable provenance are more likely to be trustworthy [10], [12], [17]. Several studies define metrics for evaluating data quality [13]–[15], assuming that data providers export and share information as proof of their honesty regarding the delivered data quality.

c) Trustworthy Data Services - Discussion: Existing work proposes models to define data quality metrics, including data freshness and performance metrics: service response time, task success ratio, and availability. Those models are often not proposed for a specific type of service. To the best of our knowledge, no trust evaluation solutions targeting precisely data services combining QoS and data quality have been proposed. There is no data trust model which seems to consider data freshness for trust evaluation. Existing solutions define data quality metrics, including data timeliness assuming that meta-data is available. In the absence of such information, there is an alternative to obtain these measures to evaluate data freshness, especially database timeliness.

III. Data Service Trust Evaluation Model

As aforementioned, services are provided under heterogeneous quality of service (QoS) conditions to support various users' needs. However, there are no guarantees that the QoS is continuously ensured. Thus, it is necessary to determine to which extent a service can guarantee a certain level of QoS. For data services, it is also necessary to determine to which extent provided data can be trusted. Data trustworthiness can be determined by measuring data quality.

A trust measure can be a representative indicator of services' QoS and data quality. Service performance and data quality can be used as a quality criterion by services' consumers for selecting services. Consequently, the trust level of a data service is a value between $[0, 1]$ defined as follows:

$$T_{DS} = \alpha \times Performance + \beta \times DataQuality \qquad (1)$$

Where the weights α, β ponder the performance and data quality factors. They depend on the preferences of target service consumers and application requirements. These factors are described in the following lines.

A. Performance Factor

The *Performance factor* measures the computing capacity of a data service through three metrics described hereafter, namely: availability, time efficiency, and task success ratio.

$$Performance = \sum W_j \times Q_j \qquad (2)$$

Where Q_j = {availability, time efficiency, task success ratio} and W_j corresponds to the weight of the metric j. The weight varies according to the importance of the metric prefered by its data service's consumer.

The general idea behind the performance factor is that data service is expected to be available when it is requested, it should adhere to the response time specified in its SLA, and it must deliver data to consumers successfully.

- **Availability (Av):** A data service is unavailable when a request is denied [7].

Thus, availability can be defined as the degree to which a data service is reachable and ready to operate when requested. Therefore, availability as a value between $[0, 1]$ defined as follows:

$$Av = \frac{A_k}{N_k} \tag{3}$$

Where A_k is the number of accepted requests by the data service k, and N_k is the total number of requests submitted to the data service k.

- **Time Efficiency (TE):** Quantifies to which extent a data service meets the expected response time (ERt) specified by its service provider.
  TE takes values between $[0, 1]$ and it is defined as follows:

$$TE = 1 - \frac{Rt}{ERt} \quad \text{if} \quad Rt < ERt \tag{4}$$

$$TE = 0 \quad \text{if} \quad Rt > ERt \tag{5}$$

  where Rt is the average response time of the service.
- **Task Success Ratio (TSR)**: Sometimes, data services may be reachable and ready to operate but unable to deliver data to consumers due, for example, to network failures.
  Thus, *TSR* measures successful data delivery in response to accepted requests. It takes values between $[0, 1]$.

$$TSR = \frac{S_k}{A_k} \tag{6}$$

  Where S_k is the number of successful requests with data delivered to destination by a data service *k*.

B. Data Quality Factor

According to [2], data quality can be evaluated using multiple data quality dimensions including *completeness, timeliness, accessibility* etc. Our work chose data freshness as a data quality indicator because our results target applications requiring up-to-date data (e.g., e-health applications).

Data freshness measures to which extent data is meaningful (i.e., recent) for a target application [16]. The principle behind this is that fresh data are more valuable and trustworthy than outdated data that lose value and negatively affect decisions.

Data freshness is measured using the notion of *timeliness* determined by two dimensions: *data timeliness* and *database timeliness*. Intuitively, data is fresh when it is up-to-date (with respect to its production time) and inserted frequently into a database. We assume that data are produced under different production rates P_R and data is fresh within a validity interval T. T is defined according to the application's domain. For instance, for our e-health scenario, we set the data validity interval to 60 seconds, meaning that temperature readings are considered fresh 60 seconds after they have been collected.

Assuming that data producers continuously send data to a data service with a specific update frequency U_f:

- ***Data Timeliness*** captures the gap between the data production time and the time when it is needed and makes sure it is still within the data validity interval $T = [t_{min}, t_{max}]$. Timeliness for data D, T_D is a value between $[0, 1]$ defined as follows:

$$T_D = 1 - \frac{t_R - t_P}{T} \quad \text{if} \quad t_R < t_{max} \tag{7}$$

$$T_D = 0 \quad \text{if} \quad t_R > t_{max} \tag{8}$$

  Where t_R represents the request time, t_P the data production time, t_{max} the maximum time for data to be fresh and t_{min} = t_P. Data feshness decreases (i.e., timeliness is lower) as T_D is closer to t_{max}. Beyond t_{max}, data is no longer fresh.
- ***Database timeliness*** (T_{DB}) measures the database update frequency. A database is updated when new data are inserted. The intuition is that frequent updates can contribute to the preservation of data freshness. This frequency must be "guessed" using a protocol based on statistical and analytics strategies for black-box services.

The above definitions of data quality metrics provide evidence to correlate data freshness metrics. For example, if a database is updated frequently, it is more likely to be timely, but not necessarily the opposite. If the database has not been updated within a data validity interval (specified by the application domain and data consumers), data is more likely to be outdated from the data consumer perspective. Thus, data quality is a value in $[0, 1]$ defined as follows:

$$DataQuality = T_D \times T_{DB} \tag{9}$$

In the next section, we present the proposed architecture DETECT.

IV. DETECT: General Description

Figure 1 shows the architecture of DETECT (Data Trust Evaluation system) that implements our model. It consists of three main modules: (i) performance measuring module (PMM), (ii) data quality measuring module (DQMM), and (iii) trust measuring module (TMM). DETECT enables data consumers to select the most reliable (i.e., trustworthy) data services according to their needs and preferences. Consumers search services and DETECT returns a list of data services ranked by their trust level. DETECT continuously updates services' trust level monitoring their behavior, computing and updating their trust measure, and tagging services with this value.

A. Performance Measurement Module

The PMM measures the data service's performance. It executes this task in two steps: monitoring and evaluation.

During the monitoring step, the *Performance Monitor* collects the performance metrics continuously using the service's API and stores performance measurements in a time-series database (TSDB). Time-series indicate the recorded response time of services in milliseconds, whether the service was available, and whether it successfully finished the job.

During the evaluation step, the *Performance Evaluator* creates a performance level base. First, it collects performance measurements from the TSDB made within a specific time

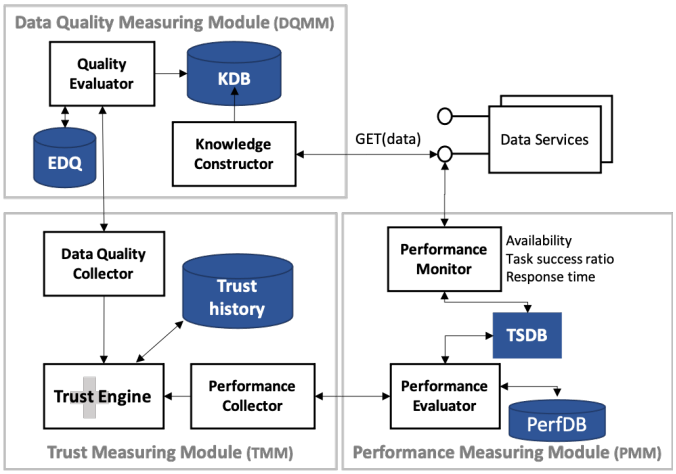

Fig. 1. Data Service Trust Evaluation Architecture.

interval. Second, it computes performance metrics as defined in III. Third, the *Performance Evaluator* evaluates the performance level of the corresponding service by applying equation 2. The *PerfDB* database stores the list of services tagged with their most recent evaluated performance level along with the evaluation timestamp.

B. Data Quality Measurement Module

DQMM observes and evaluates the quality of the data accessed by the corresponding data service using the timeliness metrics defined in section III.

The assessment of *data timeliness* and *database timeliness* can be achieved from meta-data about data quality, including data production time and database update frequency. As aforementioned, we assume that data is timestamped. However, the database update frequency is unknown to the data service's user, and the black box character of these data services hides information about the way data are captured and processed.

To overcome this problem, we propose defining and deploying a meta-data observability protocol to assess the absence of the necessary meta-data for data quality evaluation, especially the database timeliness evaluation. The protocol consists of the following steps.

First, the *knowledge constructor* observes the changes in the database state of a given data service using sampling techniques. Our protocol observes data insertions. By pulling data samples continuously with a specific sampling frequency, the *knowledge constructor* observes and measures some metrics that help to acquire knowledge about the data service's change history. The sampling frequency must be chosen considering a trade-off between retrieving enough representative samples for the computing metrics and reducing the overhead produced by sending requests and processing data. For *data sample timeliness*, we apply the equation 8 on every data item in the sample and then compute the average for all the data sets. For the *database change of state*, one would compare the different database states to observe whether the $\delta \neq \{\}$. The δ represents the intersection between two data sets belonging to two consecutive samples. Thus, the approximate update frequency of the database managed by a data service can be measured. The database *KDB* stores those metrics along with the observation's timestamp.

Second, using *KDB* during an observation period, the *quality evaluator* (1) measures the database update frequency, (2) computes the average data timeliness, and (3) uses those two metrics to evaluate the data timeliness and the data freshness of the targeted data service. Predefined T intervals of the services are stored in the DQMM. The result of this module is a list of data services tagged with their measured data quality level stored in the database *EDQ*. Data quality is measured periodically in order to keep this list up-to-date.

C. Data Service Trust Measuring Module

The Trust Measuring Module (TMM) evaluates a data service's trust level using the PMM and DQMM results. Therefore, two collectors are used in DETECT: the *performance collector* which collects performance levels of the available data services; and the *data quality collector* which collects data quality levels of the available data services. These collectors gather trust factors and feed the *trust engine* on-demand. The trust engine evaluates the trust level of each available data service using equation 1. It further enables the data requester to specify the importance of trust factors, including performance and data quality. A history of the evaluated trust levels of each service for the different requests is stored in the *trust history* which contains for every measuring timestamp the service ID, its data quality level, its performance level and the corresponding data service trust level.

V. Implementation

A prototype of DETECT implements the QoS-based trustworthy data services selection to show the feasibility of our solution. We used DETECT to set up Alice's homecare scenario for monitoring temperature. We used Prometheus[2], a monitoring tool, and its related JMETER exporters[3] to implement the *Performance Monitor*.

We implemented data services using HAPI FHIR solutions which are built from a set of components called *Resources* used to exchange and/or store data. They produce synthetic data that respect the HL7-FHIR healthcare standard and privacy constraints generated from "real" clinical/medical data. We mainly measured the quality of data accessed through the resource *Observation*[4] to support diagnosis and monitoring. Therefore, we developed our e-health scenario for monitoring temperature querying RESTful APIs. The scenario runs on a 64GB mac where data services are deployed on self-contained environments using Docker.

A. Trust case studies

We developed two case studies based on the eHealth scenario to validate the trust model and DETECT's applicability.

[2]https://prometheus.io
[3]https://jmeter.apache.org
[4]https://www.hl7.org/fhir/observation.html

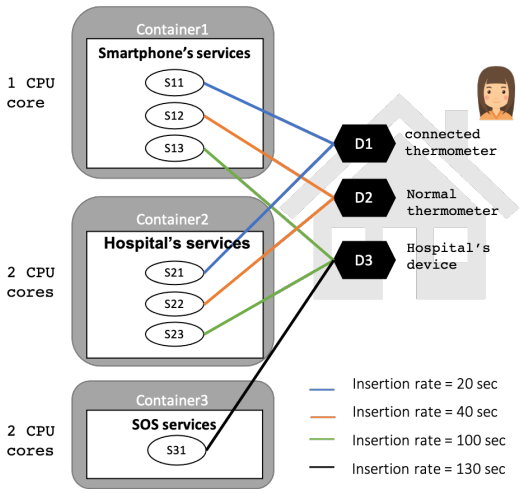

Fig. 2. Experimental setting: e-health scenario.

TABLE I
SERVICES RANKED ACCORDING TO TRUST FACTORS

Performance	*Data Quality*
S31	S11, S21
S21, S22, S23	S12, S22
S11, S12, S13	S13, S23
	S31

We assume that we have two kinds of services: on-demand services and pushed services. The chosen case studies illustrate various requirements of applications for which performance and data quality will be weighted accordingly.

- **Case Study 1: Alerting**
  This use case addresses the continuous monitoring of Alice's temperature. The devices' push service must immediately alert her doctor if it observes a significant variation in her temperature. Since Alice's safety is critical, the data must be sent timely. The performance criterion for selecting services has more priority than data freshness. Therefore, higher weight is assigned to the performance factor in the equation 1.
- **Case Study 2: Quick checkup by Alice's doctor**
  This use case assumes that Alice's doctor wants to perform a quick checkup on her health's latest indicators using on-demand services from time to time. For example, the doctor wants to have the latest temperature, and thus data must be fresh. This use case emphasizes the data quality factor (i.e., data freshness produced by services managing the data produced by the different devices). Thus, the system assigns a higher weight to the data quality factor in the equation 1.

B. Experimental Setting

In the case studies described above, three HAPI FHIR servers with different FHIR standards have been deployed on Docker containers: one simulating the hospital's server, one simulating Alice's smartphone' server, and the last one simulating the SOS server of the hospital. Each server has its independent database on the corresponding server and is reachable through its URL. To give access to these servers, we deployed 7 data services, each with its access point: three data services are deployed on the first two servers — only one service giving access to the third server. Note that these servers are configured in the same way in our controlled environment. However, to simulate the variation in the performance of the different data services, we allocated a different number of CPU cores and the different number of services to the three HAPI FHIR servers as depicted in figure 2 in a way that: (1) The bigger the number of the allocated CPU resources, the better the performance of the service since we have more resources for the server. (2) The smaller the number of the deployed services on the same container, the better the performance. The reason is that the available resources on the server are shared between a smaller number of services. For instance, we allocated 1 CPU core to the first container while allocating 2 CPU cores to the second one (see figure 2). In order to illustrate the environment of real-time real-world data services, we have simulated the process of data production and data insertion for Alice's three devices (D1, D2, D3). We have (1) set the data production rate and (2) only varied data insertion rates for all devices to have different data quality levels. For instance, device D1 has 20 seconds (sec) as an insertion rate. Table I ranks services according to the expected performance and data quality levels.

Our experiment tests the effect of α and β over the trust level of data services and their ranking. For each trust request to the *Trust Engine*, at a given time t and for a fixed performance and data quality levels, we varied α and β: we decrease the weight α from 1 to 0 while increasing β from 0 to 1 (see table II). Concerning performance evaluation at the *Performance Evaluator* level, we currently consider availability, task success ratio and response time equally important, and thus they have equal weights when computing performance level as in equation 2.

C. Results and Discussion

We performed tests using the configuration described above in order to verify our QoS-based trust model and architecture. The resulting ranked lists of services are presented in table II.

TABLE II
TRUST REQUEST OUTPUT

α=1,β=0	α=0,7,β=0,3	α=0,5,β=0,5	α=0,3,β=0,7	α=0,β=1
S31	S22	S21	S21	S21
S23	S21	S22	S22	S11
S22	S31	S31	S31	S22
S21	S23	S23	S23	S12
S13	S11	S11	S11	S31
S12	S22	S12	S12	S13
S11	S13	S13	S13	S23

According to table II, we notice that: (1) The service S31 has the highest trust level when service performance (α=1

and β=0) is preferred (higher weight). S31 is ranked lower as the performance ponder is decreased. (2) The hospital's services are ranked below S31 when α=1 and β=0. The higher is the weight pondering the data quality factor associated with a service, the higher is its trust level; in consequence, S31 is ranked before S23. (3) Services that are deployed on the first container are ranked in the lowest positions in the list when α=1 and β=0. The higher the weight that ponders the data quality factor associated with a service, the higher it is ranked. Thus, the trust level of S31 is higher than that of S13, and in consequence, S31 is better ranked. No matter the weight we attribute to trust factors, services that are deployed on the smartphone server are ranked below the services that are deployed on the hospital's server.

Hereafter, we discuss the results for each case study.

Case study 1: According to the above observations, experiments demonstrate the feasibility of our trust model and pertinence of DETECT. Indeed, as expected, results provide a suitable trust-based ranking of data services using their performance level. Note that we control the servers' performance in our configurations but not that of services deployed on the same server. These services run independently, and the allocation of resources depends on the Docker containers' load balancing and scheduling method. However, for services deployed on the same container, we can perform some actions to perturb their performance, for instance, sending more user requests.

Case study 2: In general, services have been ranked as expected. Still, S21 is better ranked than S11, S22 is better ranked than S21, and S31 is better ranked than S13 and S23. These results can be explained by the correlation between data quality and services' performance. The infrastructure configuration influences data timeliness negatively. An infrastructure configuration providing limited resources punishes the service's response time and data timeliness. This correlation affects data timeliness because, in our scenario, data changes frequently and has a short validity interval (seconds). Thus, data freshness decreases fast when it takes too long for a service to respond to a request. This dependence should be represented through the trust evaluation model defined in equation 1. Nevertheless, we did not consider this dependence yet between the trust factors in our proposed trust model.

VI. Conclusion

This paper proposes a QoS-based trust evaluation model for black-box data services combining service performance and data quality metrics. Metrics are continuously computed performing statistics on collected data. Continuous monitoring provides updated insight into data services' trust over time. Experiments showed the feasibility of the approach in applications where trust is critical.

Our future work will enhance DETECT with a data quality evaluation protocol to observe and measure the timeliness metrics for black-box services. Also, as discussed in the experimental results section, we need first to study the correlation that exists between the performance of a given data service and its data quality, and to enhance our trust model in a way to reduce the bias of the performance on the data quality factor.

Acknowledgment

This work is done in the context of the project SUMMIT number 1801172801-40892 funded by the Auvergne Rhone Alpes region AAP program.

References

[1] L. Lavieri, C. Koenig, O. Teuffel, P. Agyeman, and R.A. Ammann, "Temperatures and blood counts in pediatric patients treated with chemotherapy for cancer," NCT01683370. 2019, Sci Data 6, 108.

[2] S. Romdhani, N. Bennani, C. Ghedira-Guegan, and G. Vargas-Solar, "Trusted Data Integration in Service Environments: A Systematic Mapping," in: Service-Oriented Computing. ICSOC 2019. Lecture Notes in Computer Science, vol 11895. Springer.

[3] X. Zhao, L. Shen, X. Peng, and W. Zhao. "Toward SLA-constrained service composition: An approach based on a fuzzy linguistic preference model and an evolutionary algorithm". in Information Sciences, vol. 316, pp. 370-396, 2015.

[4] O. Jules, A. Hafid, and M. A. Serhani, "Bayesian network, and probabilistic ontology driven trust model for SLA management of Cloud services," in IEEE 3rd International Conference on Cloud Networking, 2014.

[5] C. Mao, R. Lin, C. Xu, and Q. He, "Towards a Trust Prediction Framework for Cloud Services Based on PSO-Driven Neural Network," in IEEE Access, vol. 5, pp. 2187-2199, 2017.

[6] J. Sidhu and S. Singh, "Design and Comparative Analysis of MCDM-based Multi-dimensional Trust Evaluation Schemes for Determining Trustworthiness of Cloud Service Providers," in Grid Computing, vol. 15, n. 12, pp. 197-218, 2017.

[7] P. Manuel, "A trust model of cloud computing based on Quality of Service," in Annals of Operations Research, vol. 233, n. 11, pp. 281–292, 2015.

[8] L. Qu, Y. Wang, and M. A. Orgun, "Cloud service selection based on the aggregation of user feedback and quantitative performance assessment," in 10th International Conference on Services Computing, 2013.

[9] Y. Yuyu, "A novel TOPSIS evaluation scheme for cloud service trustworthiness combining objective and subjective aspects," in Journal of Systems and Software, pp. 143, 2018.

[10] O. Hartig, "Provenance Information in the Web of Data," in Proc. of the Linked Data on the Web Workshop at WWW, 2009.

[11] C. Dai, H. S. Lim, E. Bertino, and Y. S. Moon, "Assessing the trustworthiness of location data based on provenance," In Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp. 276–285, 2009.

[12] H. S. Lim, Y. S. Moon, and E. Bertino, "Provenance-based trustworthiness assessment in sensor networks," in Proceedings of the 7th International Workshop on Data Management for Sensor Networks, pp. 2–7, 2010.

[13] T. Aamir, H. Dong, and A. Bouguettaya, "Trust in social-sensor cloud service," in 2018 IEEE International Conference on Web Services (ICWS), pp. 359–362, IEEE, 2018.

[14] H. Jin, K. Zhou, H. Jiang, D. Lei, R. Wei, and C. Li, "Full integrity and freshness for cloud data," Future Generation Computer Systems, vol. 80, pp. 640–652, 2018.

[15] S. Neumaier and J. Umbrich, "Measures for assessing the data freshness in Open Data portals," in 2016 2nd International Conference on Open and Big Data (OBD), pp. 17–24. IEEE, 2016.

[16] M. Bouzeghoub, "A framework for analysis of data freshness," in Proceedings of the 2004 international workshop on Information quality in information systems, pp. 59–67, 2004.

[17] S. Zawoad, R. Hasan, and K. Islam, "Secprov: Trustworthy and efficient provenance management in the cloud," in IEEE INFOCOM Conference on Computer Communications, pp. 1241–1249, IEEE, 2018.

[18] X. Li, J. Yuan, H. Ma, and W. Yao, "Fast and parallel trust computing scheme based on big data analysis for collaboration cloud service," IEEE Transactions on Information Forensics and Security, 13(8), pp. 1917–1931, 2018.

[19] M. Alhamad, T. Dillon, and E. Chang, "Conceptual SLA framework for cloud computing," in 4th International Conference on Digital Ecosystems and Technologies, pp. 606–610, IEEE, China, 2010.