Received: 17 January 2019 | Revised: 2 July 2019 | Accepted: 18 July 2019

DOI: 10.1002/spe.2747

EXTENDED CONFERENCE PAPER

WILEY

Service level agreement specification for end-to-end IoT application ecosystems

Awatif Alqahtani[1,2] | **Ellis Solaiman**[1] | **Pankesh Patel**[3] | **Schahram Dustdar**[4] | **Rajiv Ranjan**[1]

[1]School of Computing, Newcastle University, Newcastle upon Tyne, UK

[2]Natural and Engineering, College of Applied Studies and Community Service, King Saud University, Riyadh, Saudi Arabia

[3]Pandit Deendayal Petroleum University, Gandhinagar, India

[4]Vienna University of Technology, Vienna, Austria

Correspondence
Awatif Alqahtani, School of Computing Science, Newcastle University, Newcastle upon Tyne, UK.
Email: a.alqahtani@ncl.ac.uk

Funding information
Newcastle University

Summary

With an ever-increasing variety and complexity of Internet of Things (IoT) applications delivered by increasing numbers of service providers, there is a growing demand for an automated mechanism that can monitor and regulate the interaction between the parties involved in IoT service provision and delivery. This mechanism needs to take the form of a contract, which, in this context, is referred to as a service level agreement (SLA). As a first step toward SLA monitoring and management, an SLA specification is essential. We believe that current SLA specification formats are unable to accommodate the unique characteristics of the IoT domain, such as its multilayered nature. Therefore, we propose a grammar for a syntactical structure of an SLA specification for IoT. The grammar is built based on a proposed conceptual model that considers the main concepts that can be used to express the requirements for hardware and software components of an IoT application on an end-to-end basis. We followed the goal question metric approach to evaluate the generality and expressiveness of the proposed grammar by reviewing its concepts and their predefined lists of vocabularies against two use cases with a considerable number of participants whose research interests are mainly related to IoT. The results of the analysis show that the proposed grammar achieved 91.70% of its generality goal and 93.43% of its expressiveness goal.

KEYWORDS

IoT, service level agreement, SLA specification

1 | INTRODUCTION

The Internet of Things (IoT) is a new computing paradigm in which uniquely addressable objects such as radio-frequency identification tags, sensors, actuators, and mobile phones become part of the Internet environment.[1] IoT opens the door to innovations that will build novel types of interactions among things and humans. It enables the realization of smart cities, smart health care, and smart infrastructures and services that can enhance our quality of life and improve the utilization of resources.[2] In large-scale remote sensing applications, territorial or worldwide scope multispectral and multitemporal remote sensing data sets are utilized for data processing to meet the need for more precise and up-to-date

Abbreviations: SLA, service level agreement; SLO, service level objective; QoS, quality of service; RHMS, remote health monitoring service; EoI, event of interest; GQM, goal question metric.

knowledge.[3] The number of connected smart objects is estimated to reach 212 billion by the end of 2020.[4,5] Such a large number of connected smart objects will generate huge volumes of data that need to be analyzed and stored.[6] Storing and processing such large volumes of data are nontrivial and require the flexibility offered by cloud computing.[7] However, while the centralized architectures of cloud computing create effective economics, in the extreme case, full centralization could create unintended consequences.[8] Garcia Lopez et al[8] have mentioned four fundamental problems with centralized approaches. First, there is a need for a trade-off between releasing personal and sensitive data to use centralized services, such as social networks and location services and privacy. A second fundamental problem is that users of cloud services delegate control of the applications and systems to the cloud; this kind of delegation involves one-sided trust from clients to the clouds while blocking the launching of trust between users. A third fundamental problem is that using cloud resources neglects new generations of embedded edge devices with high computational capacity and ability to communicate seamlessly; sufficient storage space ignores that embedded intelligence power. A fourth fundamental problem is that cloud-based centralization hinders human-centered designs, which limits the link between man and machine. Therefore, moving computations to the edge under certain conditions will minimize the cloud-based centralization issues as well as enhance the utilization of edge devices' computation power. Our reference architecture for IoT is depicted in Figure 1; it illustrates the data flow across the following computing layers.[9]

1. IoT devices layer:
   This layer consists of devices that are used to sense and reflect the physical world, such as sensors, actuators, cameras, and smart mobiles.
2. Edge computing layer:
   In this layer, intelligence (computation) is pushed to the edge devices to improve performance and reduce unnecessary data transfer to cloud datacenters.
3. Cloud computing layer:
   This layer provides both infrastructure hardware as well as Big Data programming models such as stream and batch processing.

In the IoT devices layer, devices (eg, sensors, actuators, and cameras) sense, capture, and send behaviors of the physical world as raw data over computer networks to the edge layer and/or the cloud layer for further processing. The edge and cloud layers perform computational and analytical operations (eg, filtering, analyzing, detecting) on the received data to execute automatable actions on physical environments and ultimately forward visualized results to end users. The edge layer typically contains a small-scale datacenter to perform lightweight tasks. By contrast, the cloud layer consists of large, scalable, distributed pools of configurable resources for performing intensive tasks on historical and real-time data on demand.[10] The cloud layer allows users to submit jobs for computing, storing, and analysis as well as addressing the heterogeneity of data and devices.[10]

In order to ensure that the greatest benefit is derived from the IoT, it is important that service consumers and providers state their quality of service requirements within service level agreements (SLAs). "SLAs specify the contractual terms

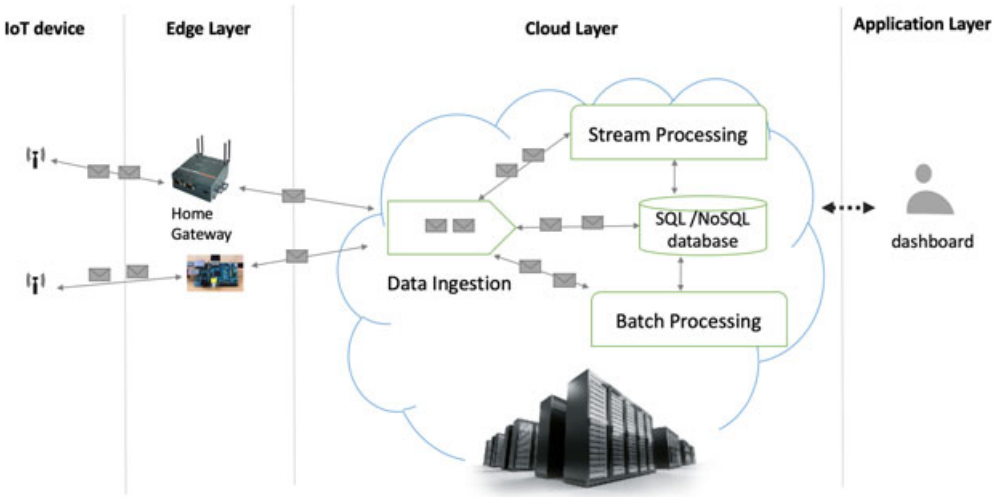


FIGURE 1 Reference architecture for the Internet of Things (IoT) [Colour figure can be viewed at wileyonlinelibrary.com]

that are used formally between consumers and providers, and they indicate the required level of service using measurable terms."[6] In the following sections, we illustrate the main characteristics of SLAs for IoT applications.

1.1 | Motivation and research problem

Many IoT applications are time sensitive. For example, let us consider a Remote Health Monitoring Service (RHMS) in which patient data is collected from different sources (eg, heartbeat sensors, smart cameras, and mobile accelerometers). The filtered data is then transferred to a large data-processing platform within the cloud layer for further analysis. One of the service level objectives (SLOs) of the RHMS might be "*detect urgent cases within* `Y` *time units and notify emergency services within* `X` *time units of detection.*" Any unacceptable delay in the transfer of the data for this application might have serious consequences. To achieve this SLO, the service provider/s should have mechanisms and guarantees in place for the availability of the service and time constraints by which any critical data must be transferred. At the very least, any hint of performance degradation or failure within the application should be monitored, investigated, and tracked to its root cause.[11] As the first step toward building IoT applications, one must be able to comprehensively specify the quality of service (QoS) constraints for an IoT application within standardized SLAs that can be understood by all stakeholders involved. To emphasize the importance of standardizing an SLA for an IoT application, consider a scenario in which an IoT application administrator would like to find the best set of providers for the services that match his/her requirements for developing the desired IoT application. Because IoT applications have a multilayered architecture, IoT administrators need to consider different categories of providers (eg, network provider, cloud provider, edge provider) and find the best candidate for each category. To be able to communicate requirements with various potential providers, it would be extremely useful to use standardized terminologies to express consumer requests as well as provider offers. Such standardization would enable the process of selecting the best candidate services to be automated. For example, the most popular cloud providers (eg, AWS, MS Azure, Oracle) currently provide take-it-or-leave-it SLAs for their services. When consumers need to compare such SLAs from different providers to select the most suitable, they need to do it manually. IoT applications can potentially be much more complex than cloud applications, and therefore such a comparison becomes more difficult. Hence, standardizing the way SLAs are described would be an important step toward automating service provider selection for both service consumers as well as service providers. In addition, providing machine readable SLAs is also an important step toward automating the process of IoT application deployment, monitoring, and dynamic reconfiguration. For example, once an IoT application has been deployed, it is important to continuously monitor the extent to which the application is adhering to what has been agreed upon as well as to reconfigure the application dynamically, on the fly, as needed to avoid/minimize SLA violations.[12] Standardizing the SLA specification for IoT applications is challenging due to a number of factors[13]:

1. the multilayered nature of IoT (IoT device layer, edge computing layer, cloud computing layer);
2. several metrics are required to capture the performance of software and hardware components of IoT applications for each layer (eg, data freshness at the edge devices and latency of stream processing at the cloud layer);
3. dependencies within each of the metrics across IoT layers (eg, data rate of stream processing at the cloud layer is affected by the sampling rate of the IoT device).

It is well known that SLA specification languages for various application domains do indeed exist, such as in other works.[14-19] However, in their current formats, to our knowledge, none of these languages are capable of accommodating the unique characteristics of the IoT domain with its multilayered nature. In other words, there is an absence of consideration for requirements of all layers (end to end) that cooperate to deliver an IoT application.

1.2 | Contributions

The main goal of this research is to design and create a new end-to-end SLA specification language for IoT while taking the challenges presented earlier into account. We summarize our contributions as follows:

- a new conceptual model that captures the knowledge base of IoT-specific SLAs by expressing the key entities of the IoT ecosystem and the relationships between those entities within the SLA context;
- a new multilayered grammar for the syntactical structure of the SLA for IoT applications;
- a tool for specifying and composing standardized end-to-end SLA constraints with a comprehensive vocabulary that provides a fine-grained level of specification to express user requirements.

The remainder of this paper is organized as follows. Section 2 introduces our IoT-based SLA conceptual model. Our SLA language and grammar are described in Section 3. Section 4 presents a graphical user interface for SLA specification using our proposed language. Section 5 reviews related work and provides a comparison of similar approaches with respect to a number of important criteria. We evaluate our work and discuss the results in Section 6. Conclusions and future research directions are presented in Section 7.

2 | AN END-TO-END SLA CONCEPTUAL MODEL FOR IOT APPLICATIONS

An end-to-end IoT ecosystem includes all components through which application data flows. Components can include hardware, software, and/or humans. From a QoS specification perspective, end-to-end SLAs should consider requirements of all of the resources (hardware and software) that cooperate to deliver the IoT application, which begins with capturing the data and ends with querying and/or storing the results of any performed analysis, in addition to any other activities, which vary depending on the use-case scenario. For example, as depicted in the conceptual model in 2, the SLA considers the requirements for all activities that are involved in the use-case scenario. To specify the requirements on an end-to-end basis, the conceptual model has considered what services (sometimes referred to within the text as software resources) are required for each activity and where the services can be deployed. Therefore, the model considers the infrastructure resources (eg, IoT devices, edge resources, and cloud resource) and where the services (eg, sensing service, real-time analysis service) can be deployed on the infrastructure resources. In the following section, we describe the concepts covered in the conceptual model, and then give a brief discussion about the relationship between the concepts. Our SLA conceptual model for IoT applications is presented in 2. In the proposed conceptual model, we refer to the reference architecture (Figure 1). The conceptual model is composed of the following entities[9]:

1. SLA: SLA includes the basic data such as the title of the SLA, its ID, type of the application (ie, smart home, smart health, etc), and start and end dates.
2. Party: Party describes an individual or group involved in the SLA and usually contains a named company or judicial entity.[20] 1. For example, in an RHMS, the parties could be the hospital management, patient, network provider, and cloud resource providers.
3. SLO: SLO provides quantitative means to define the level of service a customer can expect from a provider and expresses the intention(s) of an agreement for both the application and any involved services and infrastructure resources. SLO quantifies the required value of a QoS metric. For example, an SLO (at the application level) of the RHMS scenario could be the response to urgent cases within `Y` time units. The QoS metric in this example is `response time` and the constraint is less than `Y` time units. Furthermore, SLO parameters can be used to specify an SLO for lower level services; for example, for a data-ingestion service, an SLO can be: *ingest data with latency less than `Z` unit time*. For an infrastructure resource such as CPU of a VM, an SLO can be as follows: For an infrastructure resource such as the CPU of a VM, an SLO can be *CPU utilization is greater than 80%*.
4. Workflow Activity: IoT applications have certain activities that are required to be considered as part of the application requirements to function correctly. For example, in the RHMS, one of the possible workflow activities is capturing interesting data, analyzing real-time data, and storing interesting results in a database (eg, SQL or NoSQL).
5. Service: To achieve SLOs at the application level, it is important to establish adequate cooperation between particular services under the SLO constraints. For example, in the RHMS, to detect urgent cases within `Y` time units, it is necessary to transfer data from sensors to the ingestion service and to process data on the fly using stream-processing services. Here, we list the most common services that can cooperate to deliver SLOs of an IoT application.

    a. Sensing service: This service collects data from IoT devices and sends the collected data through a communication protocol to a higher layer. The sensing service specifies the number of sensors, types of sensors, and when to collect the data. For example, in an RHMS, a heartbeat sensor attached to the chest and an accelerometer as a hand-wrist device reflect the patient's health state continuously or periodically, based on what has been specified within the SLA for this service.
    b. Networking service: This service communicates the collected data from one layer to another. For example, in the RHMS, a home gateway uses the network to deliver collected data to the cloud for further analysis under certain bandwidth requirements.
    c. Ingestion service: This service ingests data from many data producers and then forwards them to subscribed/interested destinations such as storage and analysis services under certain requirements such as throughput limits.

d. Batch processing service: A batch-processing service receives data from resources such as ingestion layers, appends them to a master data set, and then computes batch views. For example, in the RHMS, to identify urgent cases, it is important to run machine-learning algorithms on historical patient records to recognize patterns regarding certain health issues and establish a predictive model. The predictive model can be used later with real-time data of current patients to detect particular health issues. Batch views can be computed/queried within response time constraints as specified by consumers/subscribers.
e. Stream processing service: This processes incoming data from data resources such as an ingestion service to compute real-time views. For example, collected data are processed on the fly, and if the analysis shows an abnormality such as a high heartbeat rate, then appropriate action is required, such as sending an ambulance. However, to exploit the greatest value of real-time data, consumers/subscribers can specify certain requirements such as the maximum acceptable delay for computing/querying real-time views.
f. Machine learning service: This is a service that applies different machine-learning algorithms for different purposes such as predictions as well as extracting different dimensions of knowledge from the collected facts. For example, the service may apply a machine algorithm on historical data collected from previous stroke incidents as training data to create a model for predicting a stroke based on incoming real-time data; this may save patients' lives by warning them in advance.
g. Database service (SQL and NoSQL databases): This service is used by ingestion, batch, and stream-processing services to store or retrieve data and stores data, batch views, and real-time views as intermediate or final data sets. Consumers can specify their requirements on the service such as query response time and specify whether data encryption is required.

6. Infrastructure resource: These provide the required hardware for computation, storage, and networking that are essential to deploy/run the aforementioned services. The infrastructure resources can be IoT devices, edge resources, and cloud resources.

   a. IoT device: These include devices/objects with intelligence and the ability to execute actions or reflect the physical worlds.
   b. Edge resource: These resources allow data processing at the edge network and include border routers, set-top boxes, bridges, base stations, wireless access points, and edge servers. These examples of edge resources can be equipped to support edge computation with specialized capabilities.[21]
   c. Cloud resource: These resources provide infrastructure as a service (IaaS) and are mostly located geographically far from the end devices/users.[21]

The relationships between the aforementioned entities, which are depicted in the conceptual model (Figure 2) are as follows. There is one-to-many relationship between the SLO and the SLA entities to express one or more of the QoS requirements at the application level. Therefore, each SLA entity has a composition relationship with the SLO entity. Furthermore, an SLA has a composition relationship with party since parties are responsible for providing a service, consuming a service, and/or playing third-party roles to monitor a service. An example of SLO at the application level could be *end-to-end response time of the application should be less than* `Y` *time units*. Additionally, an IoT application has a set of workflow activities (eg, capture an EoI, analyze real-time data) that cooperate to deliver the application. Therefore, there is a composition relationship between the `SLA` and `WorkflowActivity` entities. Each workflow activity requires a service (eg, sensing service, networking service, stream processing service). Each service is deployed on one of the infrastructure resources (for example, IoT device, edge resource, cloud resource). Each one of the services and infrastructure resources has one or more SLOs (eg, high level of data freshness objective for sensing service and high CPU utilization for a VM). Furthermore, each one of the services and infrastructure resources has one or more configuration metrics (eg, the sample rate of the sensing service and number of CPUs per VM of the cloud resource). Therefore, there is an association relationship between `InfrastructurResource` and `Service`, and composition relationship between `InfrastructurResource`, `Service`, `SLO`, and `ConfigurationMetric` entities. Considering an SLA on an end-to-end basis is essential because achieving the SLOs of both services and infrastructure resources has an impact on achieving SLOs at the application level. For example, in the RHMS, an SLO (SLO_{app1}) for urgent cases that require a response within less than `Y` time units is an SLO at the application level that involves many activities such as analyzing real-time data. Analyzing real-time data requires a stream-processing service that has an acceptable level of latency, and if the stream-processing service exceeds this level, then SLO_{app1} might be violated.

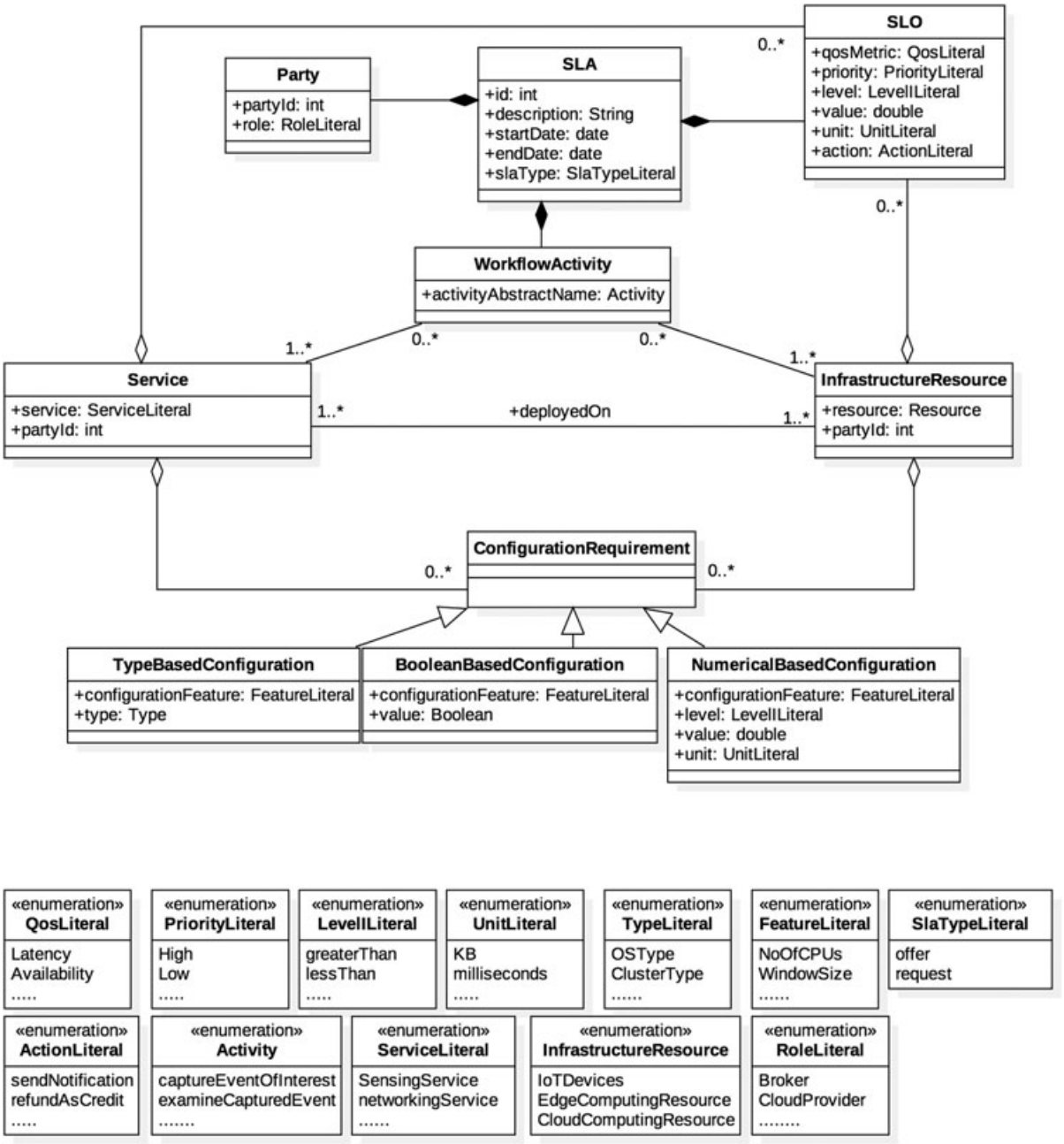


FIGURE 2 Service level agreement (SLA) conceptual model for Internet of Things (IoT) applications, capturing the key entities of an SLA and their relationships. SLO, service level objective

3 | SLA GRAMMAR FOR IOT APPLICATIONS

One of our main objectives is to provide a machine-readable SLA specification that can be used by an application orchestrator for automatically deploying IoT applications and monitoring adherence to the QoS requirements. Table 1 shows the syntax of our proposed language, which is formally defined in the extended Backus-Naur form. Some of the operators used within the grammar are as follows:

- "?": indicates that the symbol (or set of symbols in parentheses) to the left of the operator is/are optional;
- "*": means that the symbol (or set of symbols in parentheses) to the left can occur zero or more times;
- "+": means that the symbol (set of symbols in parentheses) to the left can occur one or more times.

TABLE 1 SLA Grammar of IoT Applications

Nonterminal	Production rules
< *SLA* > ::=	< *appType* >? < *slaId* >? < *startDate* >? < *endDate* >? < *description* >? < *slaType* >? < *party* > +? < *slo* >* < *workflowActivity* >*
< *workflowActivity* >::=	< *activity* > < *service* >* < *infrastructureResource* >*
< *activity* >::=	'Capture event of interest(EoI)' | 'Examine the captured (EoI) on fly' |..| 'Store Unstructured Data'
< *service* >::=	< *serviceType* > < *slo* >∗ < *configurationRequirement* >∗ < *partyId* >∗
< *serviceType* >::=	'sensingService' | 'batchProcessingService' |.....| 'machineLearningService'
< *infrastructureResource* >::=	< *infrastructureResourceType* > < *slo* >∗ < *configurationRequirement* >∗ < *partyId* >∗
< *infrastructureResourceType* >::=	'IoTDevice' | 'CloudResource' | 'EdgeResource'
< *slo* >:=	< *QoSMetric* > < *priority* > < *requiredLevel* > < *value* > < *unit* > < *action* >
< *QoSMetric* >::=	'Outage Length'|'Response Time' | 'Availability' |....| 'Timeliness' | 'Cost'
< *priority* >::=	'High'|'Medium'|'Low'
< *requiredLevel* >::=	'greater than' |'greater than or equal' | 'equal' | 'not equal' | 'less than' | 'less than or equal'
< *unit* >::=	'%' | 'millisecond' | 'seconds' |'minutes' | 'hour'| 'month' |'year' |'KB'|.....| ' per month'
< *configurationRequirement* >::=	< *booleanBasedConfiguration* > | < *typeBasedConfiguration* > | < *numericalBasedConfiguration* >
< *booleanBaseConfiguration* >::=	< *configurationFeature* > < *value* >
< *configurationFeature* >::=	'Persistence of Customer Information'| | 'Encryption Support'| 'No of vCPU' |.... … | 'Write Capacity'
< *typeBasedConfiguration* >::=	< *typeBasedConfiguration* > < *type* >
< *type* >::=	'SSD (local machine)'|.... … | 'HDD (local machine)'
< *numericalBasedConfiguration* >::=	< *configurationFeature* > < *requiredLevel* > < *value* > < *unit* >
< *party* >::=	< *partyId* > < *role* >*
< *name* >::=	< *string* >
< *role* >::=	'Cloud Provider' |'Network Provider' | 'Sensing Provider'| … | 'Broker' | 'IoT administrator' | 'End User'
< *id* >::=	< *digit* >
< *action* >::=	'sendNotification' |....| 'refundAsCredit'
< *slaType* >::=	'offer' |'request'
< *value* >::=	'string' | 'double' | 'Boolean'
< *digit* >::=	'0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'

Abbreviations: IoT, Internet of Things; SLA, service level agreement.

We have applied a context-free grammar to define the syntactic structure of the proposed SLA language for IoT. The SLA grammar consists of a list of productions; each production has a nonterminal symbol (as its left-hand side), whereas its right hand side represents the production rule of the nonterminal. The production rule consists of at least one nonterminal and/or terminal symbol. Terminal symbols are written between single or double quotes. The SLA language, then, can be produced from the given context-free grammar by simply producing a set of terminal symbols that result as an outcome of frequently replacing any nonterminal in the sequence with its production rule. For example, in the SLA grammar, SLA is a nonterminal and the production rule for the nonterminal: < *SLA* > is as follows:

< *id* > < *startDate* > < *endDate* > < *description* > < *slaType* > < *party* >+ < *slo* >* < *workflowActivity* >*.

Furthermore, within the production rules, there are many nonterminal symbols. Each of the defined symbols in the production rules of the nonterminal < *SLA* > has its own production rule. Therefore, each symbol can be replaced with its production rules. For example, < *id* > can be replaced with < *digit* >, and < *digit* > can be replaced with its production rule:

< *digit* > := '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7'| '8' | '9'.

By repeating the process of replacing each nonterminal with its production rule, the outcome is an SLA specification language for IoT. Consider the example < *SLA* > ::= < *id* > < *startDate* > < *endDate* > < *description* > < *slaType* > < *party* >+ < *slo* >* < *workflowActivity* >*. This example can be read as an SLA (< *SLA* >) that consists of an optional characteristics id (< *id* >), a date (< *startDate* >), an end date(< *endDate* >), a description (< *description* >), and an SLA type (< *slaType* >). This example additionally consists of one or more parties (< *party* >), zero or more SLOs (< *slo* >), and zero or more workflow activities (< *workflowActivity* >). One of the advantages of using a free-context grammar is to reduce misunderstandings by providing only one interpretation.

The elements < *id* >, < *description* >?, < *slaType* >?, < *party* > +?, < *startDate* >, and < *endDate* > describe basic information related to the SLA. Each SLA consists of at least one SLO < *slo* > to express the required QoS at the application level (eg, in the RHMS, `response Time` less than 2 minutes). Each SLA also contains the priority level (eg, high, medium, low) of each < *slo* >. For example, in RHMS, response time has higher priority than power consumption, which is not the case with an autolighted building for which power consumption has a high priority. The concept of a < *workflowactivity* > is used to express the data flow activities of an IoT application (eg, capture the event of interest, large-scale real-time data analysis, and large-scale historical data analysis). Each workflow activity is mapped to its required < *Service* > (such as sensing service, batch processing service) and to its < *InfrastructureResource* > (eg, IoT devices, edge resources, and cloud resources). Each service and infrastructure resource has its own < *slo* > and < *configurationMetrics* >. As mentioned earlier, the SLO can express the required QoS for each of the services. The differentiations between configuration requirements such as < *numericalBasedConfiguration* >, < *booleanBasedConfiguration* >, and < *typeBasedConfiguration* > are based on their values: some configuration features have Boolean values, others determine the type of feature, and some have numerical values. For example, number of required CPUs, encryption support, and type of cluster are examples of < *numericalBasedConfiguration* >, < *booleanBasedConfiguration* >, and < *typeBasedConfiguration* >, respectively.

4 | SLA SPECIFICATION TOOL FOR IOT APPLICATIONS

We have developed a graphical user interface (GUI)/standalone wizard that can be used by both service consumers and service providers for creating SLA requests and offers, respectively.[9,22] The tool is used to simplify the process of capturing the requirements of IoT applications and it supports the following: (1) specifying the SLOs of an IoT application at the application level; (2) specifying the workflow activities of the IoT application; (3) mapping each activity to the required software and hardware resources and specifying the constraints of SLOs and other configuration-related metrics of the required hardware and software; and (4) creating the composed SLA in JSON format. In the following sections, we present the design goals and the architecture of the tool.

4.1 | Design goals

SLA creation is an important and critical step, considering the fact that SLA-based service discovery, negotiation, monitoring, management, and resource allocation rely on what has been specified within the SLA. As a result, we have developed a toolkit that enables service consumers to specify their QoS requirements and express them as SLOs, as well as specifying some configuration-related metrics for each software/hardware component of the system. We have considered the following features as design goals of the tool[22]:

- Expressiveness: We aim to provide a rich list of domain-specific vocabularies to allow fine-grained SLA specifications.
- Generality: We aim to consider common components or layers of IoT architecture (IoT, edge, and cloud).
- Extendibility: The tool is to some extent extendable because it has been designed to allow anyone who is interested in customizing/enhancing the SLA according to his/her application-specific needs to add or delete activity/metrics without changing the programming code. It is possible to add/delete/change activity/metrics using an attached Excel file, and these changes can be reflected dynamically. The Excel file preserves the schema of SLA components (eg, workflow activities and their related software and hardware requirements).
- Simplicity: Providing a GUI enables users to specify their requirements without needing prior knowledge of a machine-readable language such as JSON or XML. Furthermore, the tool allows users to specify an SLA in the same data flow as their application by allowing users to specify the workflow activity of their application first and to then specify the requirements in the same flow of occurrences as the selected activities.

4.2 | System architecture

The abstracted design and architecture of the tool are depicted in Figure 3. The overall architecture comprises three basic layers as follows[22]:

- GUI layer, which includes the user interface components. It displays these components as a sequence of forms that guide the user through well-defined steps.

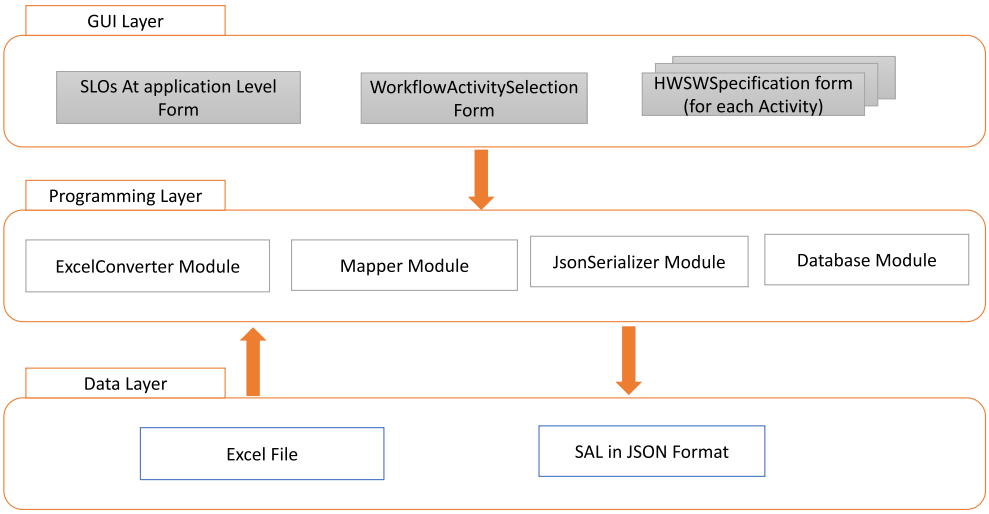


FIGURE 3 The layered architecture of the tool. GUI, graphical user interface; SLO, service level objective [Colour figure can be viewed at wileyonlinelibrary.com]

- Programming Layer, which encapsulates the programming modules in order to serve the GUI layer by providing the required functionalities.
- Data layer, which encapsulates the required data as an input to the tool or output of the tool. It includes the following:
  - An Excel file as an input, which provides data that describes the SLO and configuration metrics related to the software and hardware requirements of each activity. The Excel file has new vocabularies that provide fine-grained details regarding the associated software and hardware components of each workflow activity.
  - A JSON document as an output, which represents the SLA document. The specification is related to the SLOs at the application level followed by a specification related to each activity, which includes the SLOs and configuration-related metrics required for the programming model (eg, stream processing) and deployment layer (eg, cloud layer).

Users of the toolkit are able to perform the following steps, in sequence*:

`Step 1:` Specify the SLO constraints at the application level, such as the required/desired availability and constraints on the response time of the application (Figure 4). For example, to specify the application response time, the user can set the priority (eg, "high") and specify the required level by choosing from a drop-down menu (eg, "greater than") selecting a threshold value such as (5) and then selecting a unit such as "milliseconds".

`Step 2:` Select the workflow activities and connect them to preserve the dependency between the selected activities: a user can select the workflow activities based on his/her application scenario (Figure 5). For example, if the application is concerned with turning the heater on when the temperature is less than a specific threshold value, then the user can select activities "Capture EoI", "Examine the captured EoI", and "Actuate based on the captured event's value" and then connect them to show the sequence of the activities.

`Step 3:` Map each selected workflow activity to its required service and infrastructure resource: after selecting and connecting the workflow activities, the user can then specify, for each selected activity, the service and the infrastructure resource requirements that host the service (Figure 6). For example, the "Capture EoI" activity requires a sensing service that can be deployed on an IoT device.

`Step 4:` Specify the SLO and configuration requirements for each service and infrastructure resource: after mapping each activity to its required service and infrastructure resource, the user can begin specifying the SLO and configuration metrics for both services and infrastructure resources (Figure 7). For example, the user can specify the constraints on "data freshness" as an SLO requirement of the sensing service as well as specifying "measurement collection interval" as a configuration metric of the sensing service for "Capture EoI".

`Step 5:` Create an SLA document in JSON format: when users press the "Finish" button after specifying their requirements related to each of the selected activities, an SLA document will be generated in JSON format based on what has been specified. Figure 8 provides the basic details for the SLA of RHMS, covering the application type and start and end dates of the agreement. In addition, at line 5, it lists the SLO constraints at the application level; the snippet just shows the

*For further details regarding the tool: https://arxiv.org/abs/1810.02749.

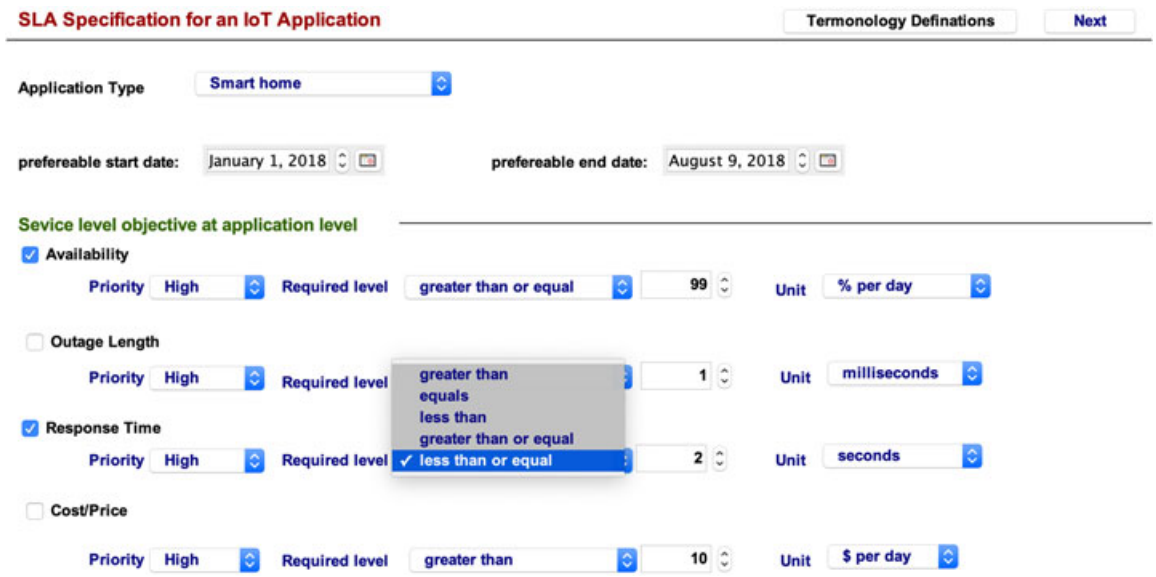


FIGURE 4 Step 1: specify service level objectives at application level. IoT, Internet of Things; SLA, service level agreement [Colour figure can be viewed at wileyonlinelibrary.com]

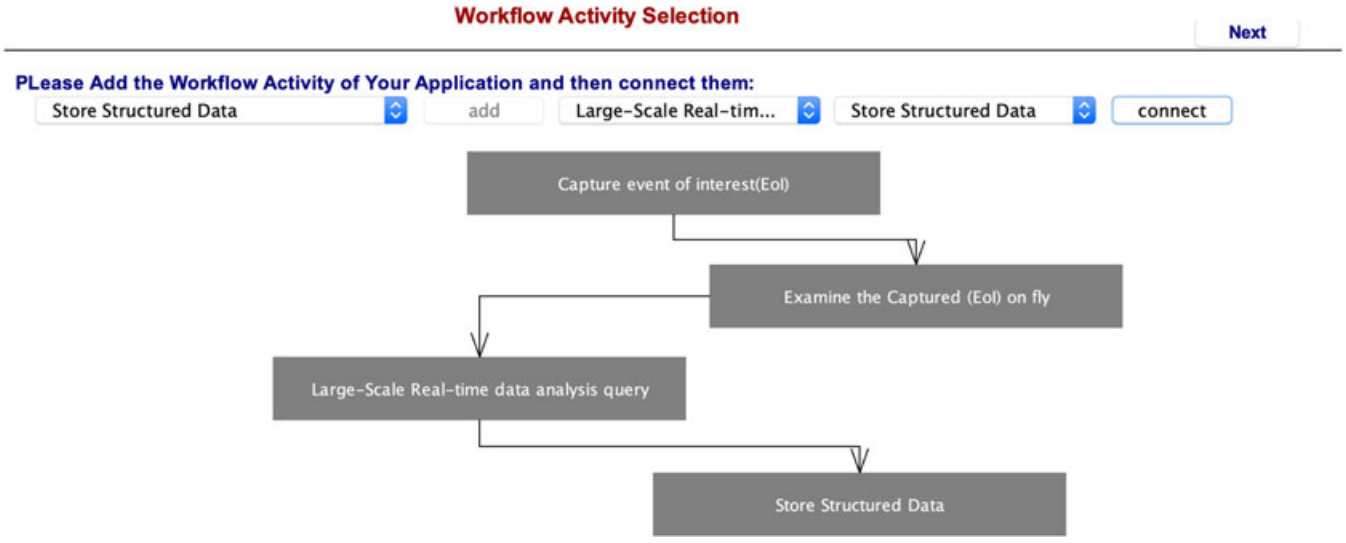


FIGURE 5 Step 2: select and connect the application workflow activities [Colour figure can be viewed at wileyonlinelibrary.com]

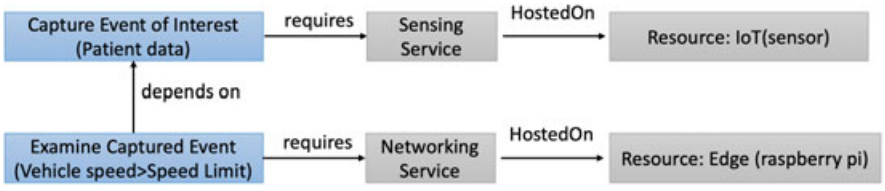


FIGURE 6 Step 3: map each selected workflow activity to its required service and infrastructure resource. IoT, Internet of Things [Colour figure can be viewed at wileyonlinelibrary.com]

`Availability` as a high priority objective with a level greater than 99% per month. The terms used within the JSON are the same terms found within the grammar. The snippet covers most of the used terms within the first line in Table 1. Some other terms listed as production rules for the nonterminal < *SLA* > such as < *description* > have not been used because they are optional.

The tool simplifies the process and guides the user through generating an end-to-end SLA and can be used to specify the requirements of different IoT applications. For example, the IoT administrator of the RHMS can specify the SLOs

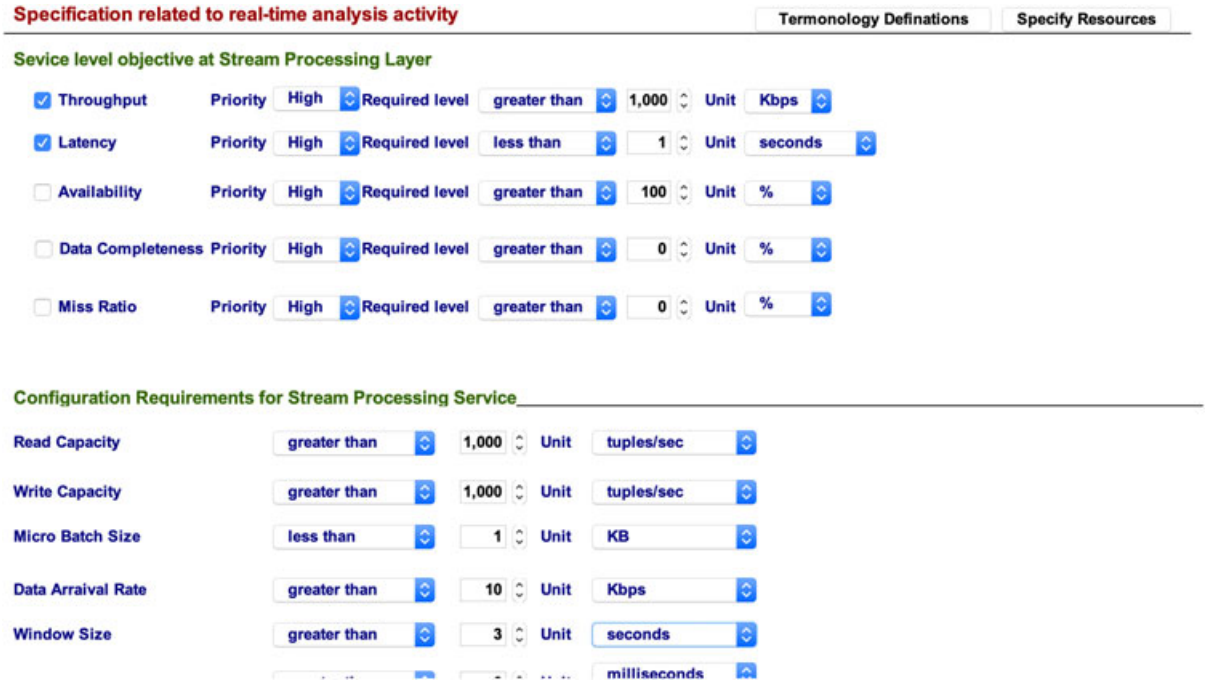


FIGURE 7 Step 4: specify the requirements of each selected activity [Colour figure can be viewed at wileyonlinelibrary.com]

```
{
  "appType" : "Smart health",
  "startDate" : "Sun Oct 21 23:05:01 BST 2018",
  "endDate" : "Mon Oct 21 23:05:01 BST 2019",
  "slo" : [ {
    "qosMetric" : "Availability",
    "priority" : "High",
    "requiredLevel" : "greater than",
    "value" : "99.0",
    "unit" : "% per day"
  } ],
  "slaid" : "Smart health Sun Oct 21 23:05:01 BST 2018Mon Oct
    21 23:05:01 BST 2019h"
}
```

FIGURE 8 Step 5: generate the service level agreement in the JSON format based on what has been specified [Colour figure can be viewed at wileyonlinelibrary.com]

of the application such as that response time to urgent cases should be less than 5 minutes. He/she will also be able to specify the activities involved such as capture EoI (eg, patients' data), examine the captured events (for filtering), analyze real-time data on the fly, and store the interesting results. Figure 9 shows the mapping process for each one of involved activities to the required service as well as the infrastructure resource. It also depicts an example of SLOs related to each one of the required services and the infrastructure resources, which are cooperating to deliver the RHMS. Figure 10 shows the abstract structure of the main concepts that are considered within the resulting SLA document with an example of each concept for clarification purposes.

5 | RELATED WORK

This paper significantly extends our previous works.[9,22] Studies related to our work can be divided into two categories. (a) Studies that attempt to identify the most important QoS metrics for one or more of the main layers of the IoT architecture are relevant because we attempt to consider the key QoS metrics within our SLA language for each of the involved layers within the reference architecture. (b) We compare studies that propose SLA languages for various applications to our own IoT SLA specification language. Section 5.1 provides the key QoS metrics that different studies have considered. They

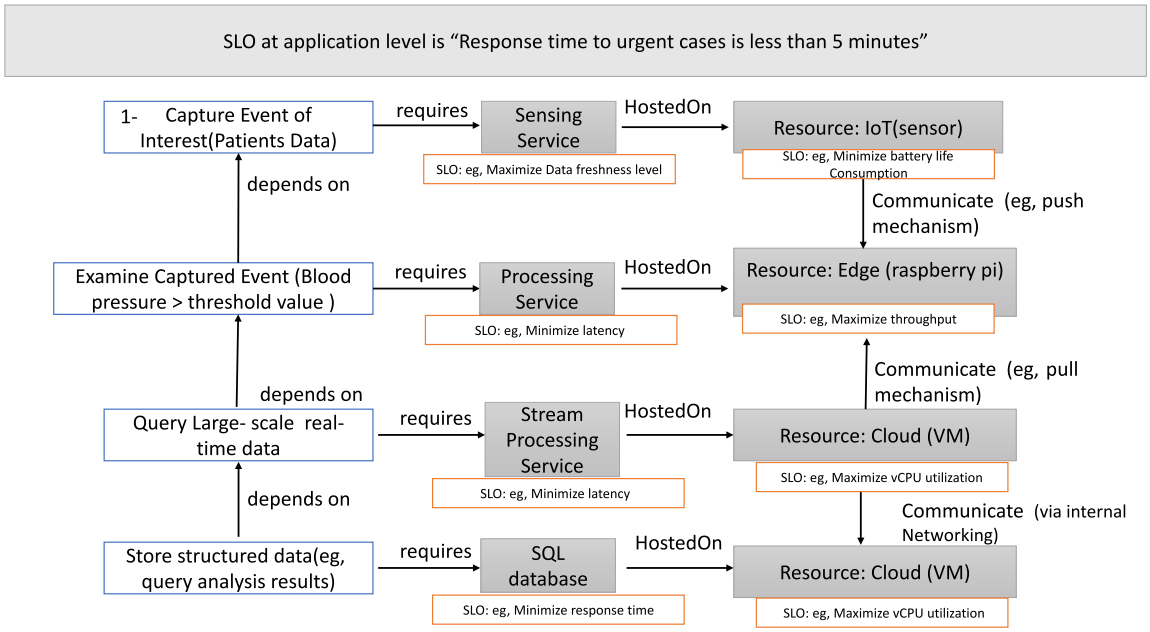


FIGURE 9 Mapping activities to the required service as well as the infrastructure resource [Colour figure can be viewed at wileyonlinelibrary.com]

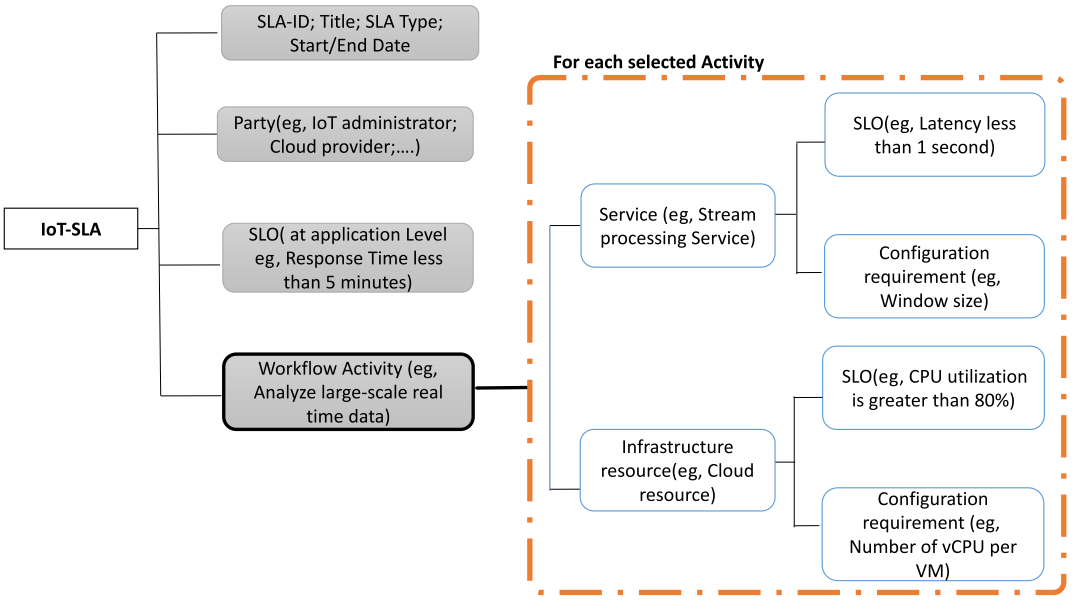


FIGURE 10 The abstract structure of the main concepts that are considered within the resulting service level agreement (SLA) document. IoT, Internet of Things; SLO, service level objective [Colour figure can be viewed at wileyonlinelibrary.com]

focus mostly on performance aspects; we review these studies to identify the most common and important QoS metrics to consider them within our proposed grammar.

5.1 | QoS metrics through the layers

Within the first class of studies, QoS metrics for the IoT device layer include the optimum number of active sensors, sensor quality, energy consumption, data volume, trustworthiness, coverage, and mobility.[23-26] Some of these metrics may have little effect when considered as a single edge device,[25] but this is not as trivial as it seems when considering the number of deployed devices that cooperate to deliver a service. For example, power consumption of 0.9 watts for a sensor seems reasonable, but when a network of hundreds of sensors is deployed, the cumulative power consumption makes a

real difference. Network layer QoS metrics, such as network availability, network capacity, and throughput, mean time to respond, mean time to repair, delay, and delay variation, are discussed in related works.[26-30] Within the cloud layer, throughput and response time are QoS requirements of the data analysis programming models, whereas CPU utilization, memory utilization, network latency, and network bandwidth are examples of QoS of the infrastructure layer.[25] The work of Uriarte et al[19] identified a list of metrics related to cloud computing infrastructures such as CPU, storage size, type of storage (eg, local storage space), the number of input/output operations on the storage specified for the service, storage bandwidth, and operating system. The work of Wang et al[31] has presented an end-to-end performance analysis that identifies key metrics impacting cloud-based topic detection and tracking applications. The analysis highlights the complexity of such applications as it captures dependencies between metrics across the cloud layers. At the IoT application layer, the key requirements vary according to the type/sector for which the IoT application is developed. For example, key requirements for health-monitoring applications are robustness, durability, accuracy, precision, reliability, security, privacy, availability, and responsiveness.[23,32,33] Low latency is a key requirement in critical and real-time applications,[10,23] whereas network utilization and energy efficiency have a high priority in less critical applications such as building automation.[23,34] In our work, we considered the most common QoS metrics for all layers of the IoT reference architecture.

5.2 | Comparison with proposed SLA languages

In the second class of studies, several projects focused on the development of SLA specification languages.[14-19] SLA*,[14] CSLA,[35] and SLAC[19] are languages that have been developed for the cloud computing paradigm. The SLA* language[14] is an abstract syntax for machine-readable SLAs and SLA templates (SLA(T)s). The SLA* language is a language that is independent of underlying technologies and can be represented by any syntactic format, such as XML or OWL. The SLA* language provides a specification of SLA(T)s at a fine-grained level of detail. SLA(T) consists of the following sections: an attribute template SLA, the parties to the agreement, service descriptions, variable declarations, and the terms of the agreement.[14] Furthermore, the language supports any kind of service and has been tested on different domains such as enterprize IT and live-media streaming. Nevertheless, specific vocabulary must be defined for each domain.[15] Moreover, the model does not support multiparty agreements.[16] Kouki et al[35] provided a cloud SLA (CSLA) to define an SLA for the cloud domain. The CSLA language consists of three sections: the validation period of the agreement, the parties of the agreement, and a reference to the template used to create the agreement. The template defines the service, constraints, the related guarantees, the billing plan, and the termination conditions.[15] The concept of fuzziness and confidence is one of the language novelties that considers the dynamic nature of cloud computing. However, there is no formalism for the SLA specification in CSLA.[15] In the work of Uriarte et al,[19] the authors proposed SLAC, which is a language to define a tailored SLA for the cloud domain. The authors specified the syntax of the language as well as the semantics to check the conformance of SLAs. However, SLAC supports only IaaS. Stamatakis and Papaemmanouil[18] presented a framework that enables application developers to specify SLA metrics, how they can be calculated, the evaluation period, and constraints to avoid SLA violations using their SLA grammar, termed XCLang. However, their main focus is the cloud database. In the work of Gamez Díaz et al,[36] the researchers proposed iAgree and SLA4OAI to define pricing plans. iAgree is a language to describe a vendor-neutral SLA and it is an open source language that aims to model a considerable number of scenarios, including computational services (eg, RESTful APIs) and human services (eg, business processes). iAgree is an active project and many enhancements have been developed since its first preliminary version. However, no studies have investigated whether it is expressive enough for IoT domain scenarios. iAgree can be a JSON or a YAML document that is, mainly, built based on a JSON-Schema. SLA4OAI is intended to model RESTful APIs. SLA4OAI is an SLA-driven Open API specification (OAS).[36] OAS document has been extended with "x-sla" as an optional attribute with a URI that points to a JSON or a YAML document representing the SLA4OAI description. Due to the limited research efforts related to an SLA specification language, specifically for IoT, we compare our proposed language with the most commonly available service contract languages of cloud and web services mentioned earlier. We use the following main criteria and present our results in Table 2.

- IoT domain: This criterion defines whether a language has been developed for the IoT domain.
- Syntax: This is supported when there is a formal definition of the syntax, eg, using BNF.
- Expressiveness: This criterion can be said to be met when the language contains domain-specific vocabulary. If it does not provide domain-specific vocabulary, then the expressiveness criterion is `partially` supported.
- Ease of use: This criterion can be viewed from the perspective of developers and service consumers. From the service consumer perspective, ease of use is achieved if the user is not required to have much knowledge about how to create the specification in a machine-readable format. From the developers' perspective, ease of use is determined by whether it is

TABLE 2 Comparison of the service level agreement (SLA) languages. Black circles represent features supported in the language, empty circles represent a partially supported feature and a hyphen (-) means not covered.[9]

Comparison Features	WSLA	WS-Agreement	SLA*	SLAng	XCLang	CSLA	SLAC	iAgree	SLA-IoT
IoT Domain	-	-	-	-	-	-	-	-	●
Syntax	●	●	●	●	●	●	○	●	●
Expressiveness	○	○	○	●	●	●	●	○	●
Ease of use	○	○	○	○	○	○	○	●	●
Support different type of computational resources	○	○	○	○	○	○	○	○	●

Abbreviations: CSLA, cloud service level agreement; IoT, Internet of Things.

written in a machine-readable format. The ease-of-use criterion is only `partially` met if just one of these perspectives has been considered.

- Support different types of computational resources: Fully supported when a language considers specification requirements of a range of resources such as IoT devices, edge resources, and cloud resources, and `partially` supported when it allows for specifying one category of required resources such as only VMs.

Although many SLA specification languages for various application domains do exist, we believe that, in their current formats, they are unable to accommodate the unique characteristics of the cloud-based IoT domain. As can be observed in our comparison Table 2, none of the compared SLA languages provide support for IoT applications. We have attempted in our specification to consider the most common/typical cloud-based IoT application layers, including data sources, the most common data analysis programming models, and computational resources (eg, IoT, edge resources, and cloud datacenters). Furthermore, there are different application models that have different stacks of essential interdependent services. For example, some applications require a certain type of data analysis programming model such as applying data ingestion and stream processing to monitor a patient's health remotely. Other applications compute statistics of a particular vehicle for a month-long period and require ingestion, stream processing, and batch processing data analysis programming models. Therefore, our SLA logic follows the workflow of IoT-based applications to simplify the process for users (eg, IoT administrators) to specify their requirements. Our SLA logic enables users to select the workflow of activities for their IoT-based applications as well as to specify their requirements for each service and its computational/storage resources (eg, specify the latency limit of the stream processing service and the number of VMs). We have developed a GUI-based tool to enable consumers to specify their requirements. The tool creates the SLA in a JSON format. By providing a GUI, we ensure the correctness of the SLA specification syntax. Most previous efforts provide the SLA template in XML format without the support of a GUI, which makes the process of creating a detailed and accurate SLA difficult.

6 | EVALUATION OF THE GRAMMAR

We encountered several problems when trying to evaluate the proposed grammar, such as the complexity of the domain and the limitation of available guidelines to evaluate the SLA specification. Therefore, we evaluated the proposed grammar by reviewing it with a number of experts within the IoT domain. After these discussions, the grammar was refined. We applied the goal question metric (GQM) approach[37] as an evaluation approach to determine whether our proposed grammar met our goals.

6.1 | Applying the GQM approach to evaluate the proposed grammar

The GQM approach specifies a number of steps to be undertaken to determine whether goals have been achieved.[37] First, the goals must be clearly specified; then, a path must be traced between these goals and the data that define them. These data can then be interpreted through a framework against the predetermined goals. Quantified information can be used to measure whether goals have been achieved.[37] The approach was initially used for assessing weakness in a set of projects in the NASA Goddard Space Flight Center environment. Despite the fact that the approach was initially utilized to characterize and assess objectives for a specific extension in a specific environment, it has been utilized to define and assist goals for a certain project within a certain environment such as the objective-setting step in an evolutionary quality-improvement model customized for a software development organization. The result of the application of the

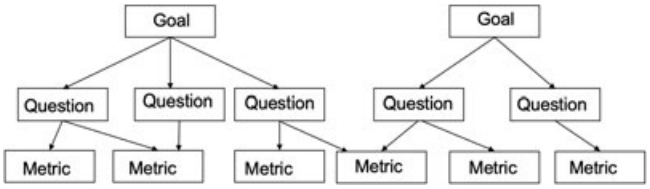


FIGURE 11 The hierarchical structure of a goal question metric model[37]

Analyze	The object under measurement
For the purpose of	understanding, controlling, or improving the object
With respect to	the quality of the object on which the measurement focuses
From the view point of	the people that measure the object
In the context of	the environment in which measurement takes place

TABLE 3 Main elements of the goal question metric goal definition template[38]

Goal 1:	Main Element from[37]	Example related to our work
	Purpose:	indicate
	Issue:	the generality
	of object:	of the proposed grammar
	view point:	from IoT experts′/IoT administrators′ viewpoints

TABLE 4 Defining our first goal following the template in the work of Caldiera and Rombach[37]

GQM approach is a measurement framework focusing on a specific set of issues and a set of rules to interpret the measured data. The measurement framework has three levels.[37]

1. Conceptual level (GOAL): In this level, a goal is defined for an object for different models of quality within a particular environment from different points of view for a variety of reasons.
2. Operational level (QUESTION): A set of questions attempts to characterize the object of measurement (product, process, resource) with regard to a chosen quality issue and to determine, as a result, its quality from that point of view.
3. Quantitative level (METRIC): This refers to quantifying the answer to a question by associating a set of data with each question.

A GQM framework consists of a hierarchical structure (Figure 11) starting with a goal. Then, the goal is refined into a set of questions to break the issue (defined within the goal) to be measured into its key components. Each question is refined into metrics as a step to quantify the questions′ answers. We believe that applying the GQM approach allows us to determine whether we have achieved our intended goals of proposing a grammar, especially because it provides a roadmap with which to measure the desired goals numerically. The GQM approach was chosen because there is a similarity between what we want to achieve and the GQM approach. We intend to check the generality and expressiveness of the proposed grammar, which can be expressed as the GOAL in the GQM approach. Under each goal, there are number of concepts that we can formulate as questions regarding them (this corresponds to Questions in the GQM approach) and calculate the metric value for each question based on participants′ answers (this corresponds to METRIC in the GQM approach). In the following section, we follow the GQM approach to define our goals and present the list of questions that will be used to calculate the metrics to reflect the percentage achieved of each stated goal.

6.1.1 | Defining the goals

We are trying to measure the generality and the expressiveness level of our proposed grammar from the viewpoints of IoT experts/IoT administrators. Therefore, we intend to specify each of the issues (generality and expressiveness of the proposed grammar) as the GOAL based on the GQM approach. GQM provides a template to define a GOAL in an unambiguous way by expressing the following main elements: purpose, perspective, and context characteristics (Table 3 illustrates the main elements of the GQM goal definition template[38]). We have set up two separate goals (see Table 4 and Table 5) following the template for designing a goal, as defined in the work of Caldiera and Rombach.[37]

6.1.2 | Defining the questions

For each of the predefined goals, a list of questions has been prepared. Answers to the questions will be quantified as a procedure to measure whether the goal has been achieved. The main purpose of the prepared questions is to check whether what has been considered within the grammar can capture the requirements for different use-case scenarios

TABLE 5 Defining our second goal following the template in the work of Caldiera and Rombach[37]

Goal 2:	Main Element from the work of Caldiera and Rombach[37]	Example related to our work
	Purpose:	indicate
	Issue:	the expressiveness level
	of object:	of the proposed grammar
	view point:	from IoT experts′/IoT administrators′ viewpoints

from the viewpoints of IoT experts/ IoT administrators. Furthermore, using free-text types of questions, the participants have an opportunity to express what else they believe is important to consider.

6.1.3 | Defining the metric

- For each question, we try to quantify the answer, calculating the user's satisfaction with whether what is in the grammar that can capture the requirements for different use-case scenarios, and what else should be considered for different parts of the concepts considered within the grammar. To calculate the satisfaction percentage for each goal, the following steps are applied.
- Count the number of missing requirements (NMRs), which can be used to express requirements from each participant's point of view. Additionally, count the number of selected requirements (NSRs) from the predefined list. Then, divide the NMRs by the number of selected requirements from the predefined list (NSR) (see Equation (1)). The result of Equation (1) reflects the missing ratio of what has been defined in regards to question$_i$ from participant$_j$ point of view. The miss ratio from a participant *j*'s point of view can be calculated as follows:

$$miss\_ratio_j = \frac{NMR}{NSR}. \tag{1}$$

  To calculate the metric value for each question, we apply equation 2 to each question$_i$:

$$Metric_i = \frac{\sum_{j=1}^{j=P} miss\_ratio_j}{p}. \tag{2}$$

  i represents the question that we are interested to calculate its metric value. $j = 1 \cdots \cdot p$, p the represents number of participants.
- Calculate overall value, which represents the satisfaction percentage of achieving the kth goal by applying the following two steps.

  1. The overall value, which represents the unsatisfied percentage of achieving the goal$_k$,

$$goal_k = \frac{\sum_{i=1}^{i=n} Metric_i}{n} * 100, \tag{3}$$

     $i = 1 \cdots \cdot n$, n represents the number of predefined questions for goal$_k$. $k = 1 \cdots \cdot G$, G represents the number of predefined goals.
  2. The overall value, which represents the percentage of satisfying goal$_k$,

$$goal_k = \left(1 - \frac{\sum_{i=1}^{i=n} Metric_i}{n}\right) * 100, \tag{4}$$

     $i = 1 \cdots \cdot n$, n represents the total number of predefined questions for goal$_k$
     $k = 1 \cdots \cdot G$, G is the total number of predefined goals.

6.1.4 | Experiment

The main purpose of the conducted experiment was to evaluate our proposed grammar to determine whether it met the predefined goals: generality and expressiveness (ie, capturing user requirements). The potential users of our proposed

work are IoT experts and administrators. Therefore, we conducted the experiment in a context in which the participants' research interests were mainly related to IoT.

6.1.5 | Procedure

The experiment was conducted following a well-defined procedure. First, a focus group discussion was held in which the participants were introduced to SLAs, to our reference architecture for IoT, to the conceptual model, and to our grammar. The participants were allowed to discuss, ask for explanations, and give instant suggestions. A use case was introduced for clarification purposes. At the end of the focus group, participants were asked to fill in a paper-pen version of a questionnaire on which there were three questions related to the first goal: indicate the generality of the proposed grammar from IoT experts'/IoT administrators' viewpoint; eleven questions were related to the second goal: indicate the expressiveness of the proposed grammar from IoT experts'/IoT administrators' viewpoint.

6.1.6 | Participants

The study was conducted with 11 participants, most of whom were researchers and PhD students working on topics related to IoT such as health care and smart-city applications. Their research interests included IoT, cloud computing, edge computing, and networking.

6.1.7 | Experiment results

The column headers of Table 6 represent the question number and the count of the vocabulary that was defined to capture requirements regarding the concept associated with each question. For example, the second column header is Q1/10, which means that 10 elements were defined related to the concept (workflow activity) (see Figure 12, which shows question 1 related to workflow activity). From participants' answers, we calculated the metric for each question. For example, to calculate the metric value for question 1 under goal 1, we followed the following steps.

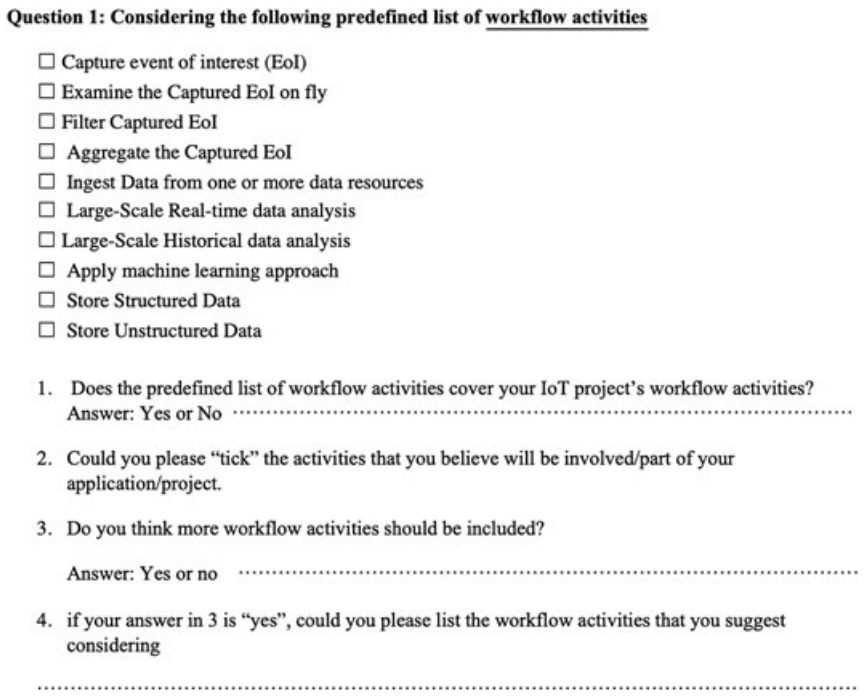
- Step1: Check the answer of the first question: the participant answered the first question, which is whether he/she thinks the predefined list of workflow activities covers his/her IoT project's workflow activities.
- Step2: Calculate the metric: Of 10 predefined activities, column 2 of Table 7 shows how many were selected by the 11 participants and column 3 shows how many more activities were suggested. Based on each participant's answer, we calculated the ratio of the missing activities from the predefined list following Equation 1 (see column 4, Table 7). Then, we calculated the corresponding metric value for question 1 following equation 1: Metric1=0.0564. Steps 1 and 2 were repeated to calculate metrics of questions related to goal 1. Questions 1, 2, and 3 were used to calculate what percentage we achieved for goal 1. Questions 4 to 14 were used to calculate the percentage we achieved for Goal 1. In other words, for each question related to goal 1, we calculated its corresponding metric following the same procedure as for question 1.
- Step3: Calculate the metric for each question: Table 6 shows the calculated metric for each question. To calculate these metrics, we followed these steps.

  1. Count the NSR that the participants agreed/believed are better to be considered. For example, participant 1 selected 9 activities from the predefined list when he/she answered the first question (Figure 7, which asks whether he/she thinks the predefined list of workflow activities covers his/her IoT project's workflow activities.
  2. Count the NMR that the participants suggested regarding the concept he/she had been asked about. For example, the first participant suggested one more activity that he/she believed should be considered as the answer to "Could you please list the workflow activities that you suggest to be considered?"
  3. Calculate the metric: Based on the participants' answers for each question, we calculated the corresponding metric for each question following Equation 1. For example, out of 10 predefined lists of activities, column 2 of Table 7 shows how many activities were selected by the 11 participants and column 3 shows how many more activities were suggested. Based on each participant's answer, we calculated the ratio of the missing activities from the predefined list following equation 1 in column 4 of Table 7 for each participant. For example, the first participant selected 9 activities from the predefined list and suggested one activity that he/she believed should be considered. Based on his/her answers, we calculated the miss ratio for this participant for question 1: NMR/NSR=1/9. Then, we calculated the corresponding metric value for question 1 following equation 1: Metric1=0.0564.

- For each predefined goal, we calculated the overall value, which represents the percentage achieving the goal. We calculated the average value of all the metrics that represented the numerical value of the questions related to each

TABLE 6 Number of selected vocabularies for each question

Participant-ID	Q1/10	Q2/3	Q3/8	Q4/8	Q5/13	Q6/20	Q7/4	Q8/10	Q9/12	Q10/5	Q11/18	Q12/15	Q13/11	Q14/12
1	6	2	4	4	8	7	1	4	6	2	8	0	3	3
2	9	3	5	6	13	16	3	9	9	5	7	8	5	2
3	10	2	4	8	11	15	4	10	4	4	7	10	6	7
4	9	2	6	8	10	16	4	10	9	4	16	13	6	11
5	8	3	7	8	13	20	4	10	12	5	18	15	11	12
6	6	3	5	5	9	12	4	9	8	0	13	9	8	9
7	10	3	8	8	13	20	4	10	12	5	18	15	11	12
8	10	3	8	8	13	20	4	10	12	5	18	14	11	12
9	7	3	7	6	12	4	4	6	4	5	14	8	7	6
10	6	2	7	7	10	19	4	8	11	4	13	11	10	11
11	10	3	8	6	13	20	4	10	0	5	13	14	8	9


Question 1: Considering the following predefined list of workflow activities

- ☐ Capture event of interest (EoI)
- ☐ Examine the Captured EoI on fly
- ☐ Filter Captured EoI
- ☐ Aggregate the Captured EoI
- ☐ Ingest Data from one or more data resources
- ☐ Large-Scale Real-time data analysis
- ☐ Large-Scale Historical data analysis
- ☐ Apply machine learning approach
- ☐ Store Structured Data
- ☐ Store Unstructured Data

1. Does the predefined list of workflow activities cover your IoT project's workflow activities?
   Answer: Yes or No .....
2. Could you please "tick" the activities that you believe will be involved/part of your application/project.
3. Do you think more workflow activities should be included?
   Answer: Yes or no .....
4. if your answer in 3 is "yes", could you please list the workflow activities that you suggest considering
   .....

FIGURE 12 Sample of the questions given to the participants. EoI, event of interest

TABLE 7 Participants' responses to question 1 as a first step to calculate the metric value of question 1 (Q1)

Participant-ID	Q1/10	Missing/suggested vocabularies for Q1
1	6	0
2	9	1
3	10	0
4	9	0
5	8	0
6	6	1
7	10	1
8	10	1
9	7	1
10	6	0
11	10	0

goal following Equation 4. For example, to calculate the overall value of the percentage achieving goal 1, there were three questions. Therefore, we calculated the average value based on the calculated metric for each question related to goal 1 (metrics of questions: Q1, Q2, Q3). Hence, the miss ratio percentage of achieving goal 1 is 8.30%, whereas the achieving percentage of goal 1 is 91.70%. Table 8 and Table 9 reflect the calculated metrics, the overall dissatisfaction and the overall satisfaction percentages for Goals 1 and 2, respectively.

6.2 | Discussion

Figure 13 displays the results based on the participants' satisfaction with the generality of our proposed grammar. The generalities of the predefined list of workflow activities and the considered computing layers were 94.36% and 92.42%, respectively, whereas the generality of the predefined services that were considered was 88.32; Figure 14 presents the results of the participants' satisfaction with the expressiveness of our proposed grammar. The expressiveness of the considered vocabularies to capture user requirements for different concepts is scored as follows: the percentage values of the expressiveness of the vocabularies for sensing, networking, and ingestion services were 80.3%, 95.05%, and 99.09%, respectively. It seems that there was a high level of satisfaction with what was considered for the networking and ingestion services, whereas there was a lower level of satisfaction with what was considered to express user requirements for the sensing service. Some participants suggested the following vocabularies to capture user requirements for the sensing service: location of device, data generation rate, type of data generated, and processing capabilities (CPU speed, memory size).

TABLE 8 Calculated metrics, overall dissatisfaction, and overall satisfaction percentage of goal 1

Participant-ID	Q1	Q2	Q3
1	0.00	0.00	0.25
2	0.11	0.00	0.60
3	0.00	0.50	0.00
4	0.00	0.00	0.17
5	0.00	0.00	0.14
6	0.17	0.00	0.00
7	0.10	0.33	0.13
8	0.10	0.00	0.00
9	0.14	0.00	0.00
10	0.00	0.00	0.00
11	0.00	0.00	0.00
Metric of each question	5.64%	7.58%	11.68%
The overall dissatisfaction percentage of goal1 :			**8.30**%
The overall satisfaction percentage of goal1 :			**91.70**%

TABLE 9 Calculated metrics, overall dissatisfaction, and overall satisfaction percentage of goal2

Participant-ID	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14
1	0.25	0.00	0.00	1.00	0.00	0.00	0.50	0.00	0.00	0.00	0.00
2	0.50	0.31	0.00	0.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3	0.00	0.00	0.00	0.25	0.00	0.00	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5	0.50	0.08	0.05	0.00	0.00	0.00	0.40	0.00	0.00	0.00	0.00
6	0.00	0.00	0.00	0.00	0.11	0.00	0.00	0.00	0.00	0.00	0.00
7	0.00	0.00	0.05	0.00	0.00	0.00	0.40	0.00	0.07	0.00	0.00
8	0.25	0.00	0.00	0.50	0.00	0.00	0.00	0.06	0.00	0.09	0.00
9	0.67	0.08	0.00	0.25	0.00	0.00	0.00	0.07	0.00	0.14	0.00
10	0.00	0.00	0.00	0.00	0.00	0.00	0.25	0.00	0.36	0.00	0.00
11	0.00	0.08	0.00	0.25	0.10	0.00	0.00	0.00	0.00	0.00	0.00
Metric of each question	19.70%	4.95%	0.91%	23.48%	1.92%	0.00%	14.09%	1.15%	3.91%	2.13%	0.00%
The overall dissatisfaction percentage of goal2:							6.57%				
The overall satisfaction percentage of goal2:							93.43%				

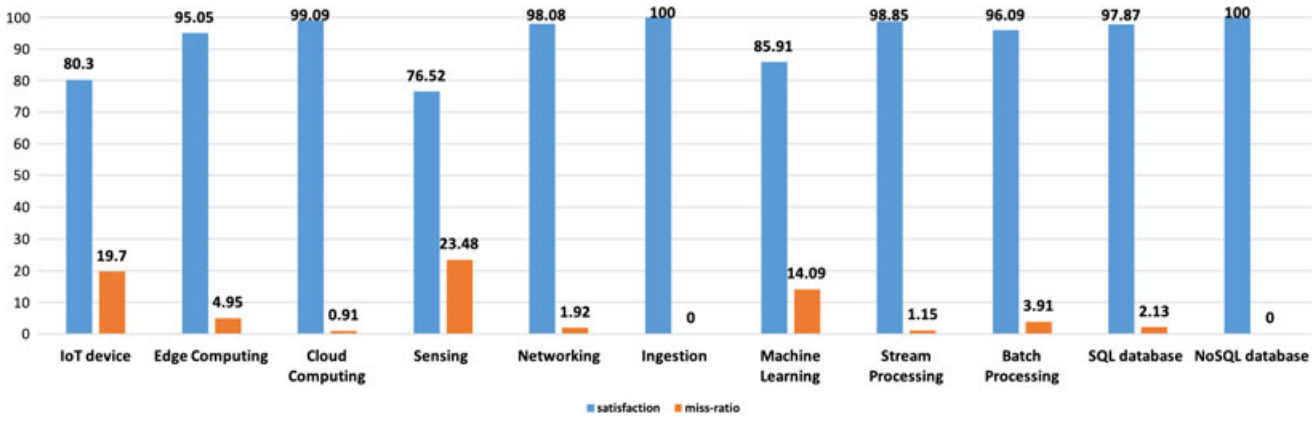


FIGURE 13 Satisfaction ratio and miss ratio for all questions related to Goal2 "Expressiveness of the grammar" [Colour figure can be viewed at wileyonlinelibrary.com]

Regarding the expressiveness of the vocabularies to capture the requirements of sensing, networking, and ingestion services, the percentage values were 76.52%, 98.08%, and 100%, respectively. There was a higher level of satisfaction with what has been considered for networking and ingestion services than for a sensing service. Some participants suggested the following vocabulary to capture user requirements for a sensing service: data type (temperature, humidity), unit of measurement, and data quality. Regarding machine learning, stream processing, batch processing, the SQL database, and the NoSQL database services, the percentage values of the expressiveness of the vocabularies used for each were

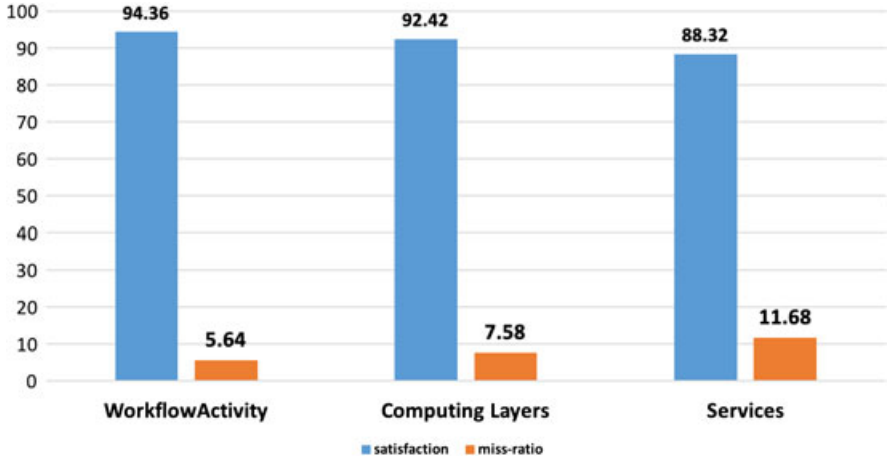


FIGURE 14 Satisfaction ratio and miss ratio for all questions related to Goal1 "Generality of the grammar" [Colour figure can be viewed at wileyonlinelibrary.com]

85.91%, 98.85%, 96.09%, 97.87%, and 100%, respectively. There was a lower level of satisfaction with what was considered for machine learning. Some participants suggested that the following vocabularies be used to capture user requirements regarding machine learning: type of machine learning classification, training and prediction for real-time or batches, and feature extractor. Table 8 and Table 9 present the calculated metrics for each question based on each participant's answer. They also present the overall percentages for each goal, showing to what degree, in percentages, we have achieved our goals. We were striving to achieve above 75% for each goal. As depicted in Tables 8 and Table 9, we achieved 91.70% of our goal to provide a general grammar for different IoT applications and 93.43% of our goal to provide an expressive grammar to capture user requirements.

6.3 | Validity concerns

A concern regarding the validity of this study is the small sample size (11 participants) because we were looking for participants whose research interests were related to IoT. The small sample size might affect the overall accuracy of the study but could be mitigated by the fact that we sought review of the proposed work from experts in the field.

7 | CONCLUSION AND FUTURE WORK

The development of an automated end-to-end IoT SLA authoring mechanism that considers the system requirements of software and hardware components, their related constraints, their interdependencies, and that is machine readable can play a significant role in automating the deployment, monitoring, and dynamic reconfiguration of IoT applications. The probability of SLA compliance with no need for human intervention is increased by providing an SLA-aware monitoring system. We believe that a machine-readable SLA can be used as a roadmap for system architects and developers. Defining "SLA offers" and "SLA requests" using standard vocabularies eases the process of comparing available options and of selecting the most suitable SLA offer based on consumer requirements. To this end, we have proposed an SLA specification that reflects the workflow activities of an IoT application and their related requirements in an unambiguous way. Our approach to specifying and composing an end-to-end SLA for IoT applications consists of three main phases. First, a conceptual model represents a knowledge base of SLA specification and composition by capturing the key entities of an SLA and their relationships. Second, a syntax grammar for end-to-end SLAs is derived from the proposed conceptual model. Third, a tool provides a GUI that allows a user to specify SLAs based on the workflow activity of an IoT application; the tool then produces an SLA in a JSON format. Lastly, we applied the GQM approach to evaluate the generality and expressiveness of the proposed grammar.

In addition to the application scenarios discussed in the introduction, our proposed SLA language can form a core component of a number of different application areas. For example, we already see the impact of our work in the development of "IoT-CANE" (Context-Aware recommendatioN systEm).[9,39] IoT-CANE is a rule-based recommender system that suggests configuration knowledge artifacts for resources across layers (edge/cloud) of IoT applications. In its processing layer, IoT-CANE uses our language and SLA specification tool to create the base knowledge of the recommender for different possible configurations, which are later selected from depending on consumer requirements. The created knowledge is stored in a configuration knowledge database. Another important application area of our work is in the development

of SLA aware algorithms for scheduling IoT workflow activities across edge and cloud resources while maximizing QoS under cost constraints. Such SLA aware algorithms are only possible if all the essential components of the end-to-end IoT ecosystem are considered within the SLA language, which is an important feature of our contribution.

As part of our future work, we are aiming to represent the knowledge base of our conceptual model as an ontology. Furthermore, we will develop an SLA-based broker system for IoT applications. The aim of the SLA-based broker system is to receive the generated machine-readable SLA (SLA offers and SLA requests) and find the best candidate that matches user requirements as a step for automating service provider selection.

We are also in the process of implementing a contract monitoring mechanism capable of interpreting SLAs generated using our proposed grammar. The aim is to automatically monitor that agreed QoS between service providers and service consumers is delivered according to the agreed SLA. To this end, an SLA deployed to our monitoring mechanism will act as a "smart contract" that logs important events such contract breaches securely for any future dispute resolution. A key technology we are considering for implementing this smart contract monitoring mechanism is Blockchain, as discussed in more detail in the works of Molina-Jimenez et al.[40,41] An important benefit of using Blockchain technology to monitor and enforce IoT SLA smart contracts is that the monitoring process could be conducted transparently by all IoT application participants. This would mitigate the need for a third party to act as a monitoring service, which may, in some circumstances, be biased to one of the interacting parties. Furthermore, through its consensus and security mechanisms, Blockchain provides a platform to ensure that agreed-upon SLA terms and any logged interactions are secured and unalterable.

ACKNOWLEDGEMENT

This research is funded and supported by Newcastle University, Newcastle, United Kingdom, and King Saud University Riyadh, Saudi Arabia.

ORCID


Awatif Alqahtani https://orcid.org/0000-0002-9794-6390

REFERENCES

1. Flammini A, Sisinni E. Wireless sensor networking in the internet of things and cloud computing era. *Procedia Engineering*. 2014;87:672-679.
2. Buyya R, Dastjerdi AV. *Internet of Things: Principles and Paradigms*. Cambridge, MA: Elsevier; 2016.
3. Wang L, Ma Y, Yan J, Chang V, Zomaya AY. pipsCloud: high performance cloud computing for remote sensing big data management and processing. *Future Gener Compute Syst*. 2016;78:353-368.
4. Gantz J, Reinsel D. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. *IDC iView: IDC Anal Future*. 2012;2007(2012):1-16.
5. Galati A, Djemame K, Fletcher M, Jessop M, Weeks M, McAvoy J. A WS-agreement based SLA implementation for the CMAC platform. In: *Economics of Grids, Clouds, Systems, and Services*. Cham, Switzerland: Springer International Publishing; 2014:159-171.
6. Radha K, Rao B, Babu S, Rao K, Reddy V, Saikiran P. Service level agreements in cloud computing and big data. *Int J Electr Comput Eng*. 2015;5(1):158.
7. Zheng X, Martin P, Brohman K, Xu LD. Cloud service negotiation in internet of things environment: a mixed approach. *IEEE Trans Ind Inform*. 2014;10(2):1506-1515.
8. Garcia Lopez P, Montresor A, Epema D, et al. Edge-centric computing: vision and challenges. *SIGCOMM Comput Commun Rev* 2015;45(5):37-42. http://doi.acm.org/10.1145/2831347.2831354
9. Alqahtani A, Li Y, Patel P, Solaiman E, Ranjan R. End-to-end service level agreement specification for IoT applications. Paper presented at: 2018 International Conference on High Performance Computing & Simulation (HPCS); 2018; Orleans, France.
10. Díaz M, Martín C, Rubio B. State-of-the-art, challenges, and open issues in the integration of Internet of things and cloud computing. *J Netw Comput Appl*. 2016;67:99-117.
11. Solaiman E, Ranjan R, Jayaraman PP, Mitra K. Monitoring internet of things application ecosystems for failure. *IT Professional*. 2016;18(5):8-11.
12. Ranjan R, Garg S, Khoskbar AR, Solaiman E, James P, Georgakopoulos D. Orchestrating bigdata analysis workflows. *IEEE Cloud Comput*. 2017;4(3):20-28.
13. Alqahtani A, Solaiman E, Buyya R, Ranjan R. End-to-end QoS specification and monitoring in the internet of things. *IEEE Tech Comm Cybern Cyber-Physical Systems*. 2016:9-13.
14. Kearney KT, Torelli F, Kotsokalis C. Sla⋆: An abstract syntax for service level agreements. Paper presented at: 2010 11th IEEE/ACM International Conference on Grid Computing; 2010; Brussels, Belgium.

15. Uriarte RB. *Supporting Autonomic Management of Clouds: Service-Level-Agreement, Cloud Monitoring and Similarity Learning* [Phd thesis]. Lucca, Italy: IMT School for Advanced Studies Lucca; 2015.
16. Maarouf A, Marzouk A, Haqiq A. A review of SLA specification languages in the cloud computing. Paper presented at: 2015 10th International Conference on Intelligent Systems: Theories and Applications (SITA); 2015; Rabat, Morocco.
17. Andrieux A, Czajkowski K, Dan A, et al. Web services for management (ws-management) specification. Paper presented at: Open Grid Forum; 2007; Seattle, WA.
18. Stamatakis D, Papaemmanouil O. SLA-driven workload management for cloud databases. Paper presented at: 2014 IEEE 30th International Conference on Data Engineering Workshops; 2014; Chicago, IL.
19. Uriarte RB, Tiezzi F, Nicola RD. SLAC: A formal service-level-agreement language for cloud computing. In: 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing; 2014; Washington, DC.
20. Gaillard G, Barthel D, Theoleyre F, Valois F. SLA specification for IoT operation-the WSN-SLA framework. 2014.
21. Mahmud R, Kotagiri R, Buyya R. Fog computing: a taxonomy, survey and future directions. In: *Internet of Everything*. Singapore: Springer Nature Singapore Pte Ltd; 2016.
22. Alqahtani A, Patel P, Solaiman E, Ranjan R. Demonstration abstract: A toolkit for specifying service level agreements for IoT applications. 2018. arXiv preprint arXiv:1810.02749.
23. Fok CL, Julien C, Roman GC, Lu C. Challenges of satisfying multiple stakeholders: Quality of service in the internet of things. In: Proceedings of the 2nd Workshop on Software Engineering for Sensor Network Applications; 2011; Honolulu, HI.
24. Calbimonte JP, Riahi M, Kefalakis N, Soldatos J, Zaslavsky A. Utility metrics specifications. openiot deliverable d422. 2014.
25. Jayaraman PP, Mitra K, Saguna S, Shah T, Georgakopoulos D, Ranjan R. Orchestrating quality of service in the cloud of things ecosystem. Paper presented at: 2015 IEEE International Symposium on Nanoelectronic and Information Systems; 2015; Indore, India.
26. Duan R, Chen X, Xing T. A QoS architecture for IoT. Paper presented at: 2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing; 2011; Dalian, China.
27. Li B, Yu J. Research and application on the smart home based on component technologies and internet of things. *Procedia Engineering*. 2011;15:2087-2092. CEIS 2011.
28. Practical guide to cloud service agreements version 2.0. 2015. http://www.cloud-council.org/deliverables/CSCC-Practical-Guide-to-Cloud-Service-Agreements.pdf. Accessed March 31, 2018.
29. Kim EC, Song JG, Hong CS. An integrated CNM architecture for multi-layer networks with simple SLA monitoring and reporting mechanism. Paper presented at: NOMS 2000. 2000 IEEE/IFIP Network Operations and Management Symposium 'The Networked Planet: Management Beyond 2000" (Cat. No.00CB37074); 2000; Honolulu, HI.
30. Bhuyan B, Sarma HKD, Sarma N, Kar A, Mall R. Quality of service (QoS) provisions in wireless sensor networks and related challenges. *Wirel Sens Netw*. 2010;2(11):861.
31. Wang M, Jayaraman PP, Solaiman E, et al. A multi-layered performance analysis for cloud-based topic detection and tracking in big data applications. *Future Gener Comput Syst*. 2018;87:580-590. http://www.sciencedirect.com/science/article/pii/S0167739X17315935
32. Vermesan O, Friess P, Guillemin P, et al. Internet of things beyond the hype: Research, innovation and deployment. 2015.
33. Savola RM, Savolainen P, Evesti A, Abie H, Sihvonen M. Risk-driven security metrics development for an e-health IoT application. Paper presented at: 2015 Information Security for South Africa (ISSA); 2015; Johannesburg, South Africa.
34. Moreno MV, Úbeda B, Skarmeta AF, Zamora MA. How can we tackle energy efficiency in IoT based smart buildings? *Sensors (Basel, Switzerland)*. 2014;14(6):9582-9614.
35. Kouki Y, de Oliveira FA, Dupont S, Ledoux T. A language support for cloud elasticity management. Paper presented at: 2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing; 2014; Chicago, IL.
36. Gamez Díaz A, Fernández Montes P, Ruiz Cortés A. Fostering SLA-driven API specifications. Paper presented at: XIV Jornadas de Ciencia e Ingeniería de Servicios (JCIS 2018); 2018; Seville, Spain.
37. Caldiera VRBG, Rombach HD. Goal question metric paradigm. *Encycl Softw Eng*. 1994;1:528-532.
38. van Solingen DR, Berghout EW. *The Goal/Question/Metric Method: A Practical Guide For Quality Improvement of Software Development*. London, UK: McGraw-Hill; 1999.
39. Li Y, Alqahtani A, Solaiman E, et al. IoT-CANE: A unified knowledge management system for data-centric internet of things application systems. *J Parallel Distributed Comput*. 2019;131:161-172.
40. Molina-Jimenez C, Solaiman E, Sfyrakis I, Ng I, Crowcroft J. On and off-blockchain enforcement of smart contracts. Paper presented at: European Conference on Parallel Processing; 2019; Göttingen, Germany.
41. Molina-Jimenez C, Sfyrakis I, Solaiman E, et al. Implementation of smart contracts using hybrid architectures with on and off-blockchain components. Paper presented at: 2018 IEEE 8th International Symposium on Cloud and Service Computing (SC2); 2018; Paris, France.

How to cite this article: Alqahtani A, Solaiman E, Patel P, Dustdar S, Ranjan R. Service level agreement specification for end-to-end IoT applications ecosystems. *Softw Pract Exper*. 2019;1–23. https://doi.org/10.1002/spe.2747