Monitoring Cloud Services through Models at Runtime: A Case in an Ambient Assisted Living Environment

Priscila Cedillo[*], **Emilio Insfran**[†], **and Silvia Abrahão**[†]
[*]Universidad de Cuenca, Ecuador
[†]Universitat Politècnica de València, Spain

ABSTRACT Ambient Assisted Living (AAL) has become an important domain that provides software systems and services to support and improve people's daily quality of life. Yet, it has not achieved large market penetration. Existing works suggest that this is primarily due to not sufficiently addressing the quality requirements posed by healthcare organizations. In particular, there is a lack of comprehensive frameworks that allow the assessment of the quality of AAL systems. In previous work, we introduced a method for monitoring cloud services using models at runtime (MoS@RT), which allows the monitoring requirements or the metric operationalizations of these requirements to be changed at runtime without the modification of the underlying infrastructure. The method and its supporting infrastructure have been successfully applied in the monitoring of cloud services, but further evaluation in other domains is needed. In this paper, we report a case study on the use of MoS@RT to monitor cloud services in an AAL environment aimed at supporting the elderly's well-being. The results suggest that relevant quality attributes of AAL systems can be adequately monitored using MoS@RT and that the report generated by the monitoring infrastructure is useful for service providers and customers to help them ensure that cloud services meet the required levels of quality.

KEYWORDS Software as a Service, Quality of Service, Services Monitoring, Ambient Assisted Living, Models at Runtime, Case Study

1. Introduction

Cloud Computing represents a fundamental shift in how cloud applications should be built, deployed, executed, and monitored. Aiming to be competitive in the current economy, organizations, especially Small and Medium Enterprises (SMEs), should be able to quickly offer cloud services ensuring a high-level of quality meanwhile satisfying the customers changing needs. However, new challenges have arisen that need to be addressed, the most critical being the provision of adequate and high-quality services that the provider must offer to its customers. These quality requirements make those who contract services in the cloud decide to negotiate with one or another provider that best meets their needs (e.g., high levels of availability and performance), which need to be conveniently measured (Guerron et al. 2020). The quality requirements of services deployed on the cloud are reflected in Service Level Agreements (SLAs) that describe both the commitments between the service provider and its customers and the penalties to be applied in the event of non-compliance with those agreements (Baset 2012). In this context, it is necessary to know the service provisioning behavior to check the cloud services quality (Singh et al. 2020).

Given the need to know such behavior, several methods and tools have been proposed to monitor the way services are being provided to customers. However, many of these solutions are not well suited to customer monitoring needs related to high-level business goals and platform-specific resources. In particular, these solutions have focused on some specific quality characteristics (e.g., performance), the monitoring of low-level quality attributes (e.g., CPU, memory, and disk usage), and some of them lack mechanisms to aggregate multiple quality attributes or parameters for a service consumer, which is a critical aspect of monitoring. In addition, the deployment and execution of software systems in highly dynamic infrastructures (i.e., clouds) lead to a new set of challenges and requirements with

JOT reference format:
Priscila Cedillo, Emilio Insfran, and Silvia Abrahão. *Monitoring Cloud Services through Models at Runtime: A Case in an Ambient Assisted Living Environment*. Journal of Object Technology. Vol. 21, No. 04, 2022.
 http://dx.doi.org/10.5381/jot.2022.21.04.a1

regard to monitoring. A monitoring system should consequently have the flexibility to be adapted or changed according to new monitoring requirements and should keep up when a service scales up or down dynamically (Katsaros et al. 2012).

Motivated by these limitations, in previous work, we presented a method for monitoring cloud services using models at runtime (MoS@RT), which allows the monitoring requirements or the metric operationalizations of these requirements to be changed at runtime without the modification of the underlying infrastructure (Cedillo et al. 2015, 2016). This is thanks to the dynamic reflection mechanisms of models at runtime which allows the measurement function of a metric to be changed in order to satisfy quality requirements as well as to define new quality requirements at runtime.

The method and its supporting infrastructure (Cloud MoS@RT) have been successfully applied in the monitoring of SaaS for supermarkets that have been deployed on the Microsoft Azure (Cedillo et al. 2016) and Google App Engine (Abrahão & Insfran 2017) platforms. We have also reported the results of an empirical study aimed to evaluate the perceived efficacy of 58 undergraduate students when using the infrastructure to configure the monitoring of cloud services (Cedillo et al. 2021). However, further evaluation of MoS@RT and its infrastructure in other domains is needed.

In particular, cloud computing interacts with other technologies such as the Internet of Things (IoT), where "a network of devices -each one embedded with sensors- are connected to the Internet" (Liau et al. 2006). This infrastructure of services and devices has allowed dealing with problems from various domains in recent years, including the Ambient Assisted Living (AAL) domain, which provides software systems and services to support and improve people's daily quality of life. However, AAL has not yet achieved large market penetration. Existing works suggest that this is primarily due to not sufficiently addressing the quality requirements posed by healthcare organizations (Garcés et al. 2017). In addition, there is a lack of a comprehensive framework that allows assessing the quality of AAL systems (Walderhaug et al. 2012).

In this paper, we, therefore, report the results of a case study aimed at assessing the feasibility of Cloud MoS@RT to monitor the quality of cloud services in an AAL environment. During the execution of the case study, the collected data have been stored in a private cloud, using a Fog Computing infrastructure (Iorga et al. 2018). Next, data and applications interact with a public cloud platform, which allows data storage and actions requiring a high level of service quality. That platform triggers events that enable adequate management of medical emergencies in AAL environments and therefore improve the storage and utilization of cloud resources.

This paper is structured as follows. Section 2 discusses existing methods and tools to monitor cloud services. Section 3 provides an overview of MoS@RT and its monitoring infrastructure. Section 4 describes the application of our monitoring approach to an AAL environment. In particular, we describe the design, execution and data analysis of the case study. Finally, Section 5 presents our conclusions and further work.

2. Related Work

Cloud computing is considered a technological paradigm that leverages technologies such as the Internet of Things (IoT). IoT and hence cloud computing has been integrated into ambient intelligence solutions and its sub-areas, including AAL. AAL's primary objective is to support people to achieve as independent a life as possible through interconnected and integrated sensors and devices in care environments (Cedillo et al. 2018). It is aimed in most cases at people with disabilities and elderly. Data scientists can further analyze the vast amount of data generated within AALs and available to healthcare personnel. In this context, the quality of services and data of these systems are crucial to support the control of people's health and well-being. Therefore, over the last years several monitoring methods and tools have been proposed to allow the most appropriate corrective and preventive actions to be taken.

In the following subsections, we analyze these solutions and their limitations. It is also necessary to reflect on the possible existing challenges in monitoring cloud systems and how, according to technological evolution, our proposed solution can scale and be applied to different domains and emerging fields. Therefore, we also discuss AAL challenges and emerging domains in cloud environments that may affect the way cloud services should be monitored.

2.1. Approaches for monitoring services on the cloud

Several approaches and tools to monitor cloud services have been proposed over the last years. For example, Singh et al. (Singh et al. 2020) proposed STAR as a solution to assess the quality of cloud services. The approach focuses on reducing the number of quality violations for efficient delivery of services. This solution is based on the MAPE-K loop, and the service quality is specified as part of the non-functional requirements to be monitored. This information is collected through sensors and then transferred for data analysis.

In other studies, the authors proposed an architecture for monitoring cloud applications called CASViD (Emeakaroha et al. 2012). This architecture monitors and detects defects related to the quality of services and resource allocation and deployment tools at the application level. However, it does not allow changing the requirements and/or metrics at runtime. Muller et al. (Muller et al. 2014) have designed and implemented SALMonADA, which is a service-based system for monitoring and analyzing the quality of services. The quality requirements is described using a monitoring management document (MMD) to be consumed by the monitoring infrastructure; however, the approach do not help stakeholders in selecting alternative measurement functions depending on the platform, and require advanced users with knowledge about metrics and specific platforms.

On the other hand, (Modi et al. 2018) presented an approach to managing and monitoring cloud services based on ontologies. In this study, the broker is the one who monitors compliance with the quality requirements. An algorithm is further proposed for provisioning services based on predictions. When a quality requirement is not fulfilled, an alert is sent to service providers and users, applying the algorithm based on the fore-

cast. This algorithm finds the virtual machine with the most negligible load and relocates the service to that virtual machine. Although this approach monitors cloud services, this solution is mainly oriented toward brokers. Therefore, this approach is not flexible enough to include all stakeholders' perspectives on monitoring needs. (Shatnawi et al. 2018) proposed an approach called CloudHealth, which evaluates the health of cloud services. This solution supports three main activities (i.e., configuration, deployment, and operation) and allows evaluating high-level monitoring goals by mapping quality characteristics, metrics, and probes. However, its approach needs to repeat the configuration and deployment for new monitoring goals or changes.

There are some approaches that are based on agents. For example, (Alhamazani et al. 2019) proposed a framework for quality monitoring of cloud applications distributed through cloud layers (*-aaS) and distributed among multiple cloud providers. The proposal provides the ability to monitor and benchmark the quality of individual application components (e.g., databases, web servers) deployed on heterogeneous public or private clouds. However, the framework employs an agent-based approach to monitor application components that collect and send specific QoS values requested by a global handler. A particular monitoring agent is defined for each cloud provider, which is not flexible enough to include, remove or change monitoring requirements at runtime. Similarly, (Lu et al. 2016) proposed a monitoring system for cloud platforms based on agents. This system, called JTangCMS, collects, delivers, and processes information from different entities. For data delivery, a framework is implemented to transfer a large amount of information at runtime. For data processing, a cloud platform is implemented to support decision-making related to cloud management. However, agents are specifically implemented to extract low-level information available from each cloud platform. The agents must be redefined to access different counters or APIs from external tools.

Furthermore, many cloud providers enable their consumers to monitor cloud services using available monitoring tools for CPU, storage, and network (Alhamazani et al. 2015). These tools are often closely integrated with their platforms. For example, CloudWatch (offered by Amazon) is a monitoring tool that enables consumers to manage and monitor their applications that reside in AWS EC2 (CPU) services. However, this monitoring tool does not have the ability to monitor a service component that may reside in the infrastructure of other cloud providers such as GoGrid and Azure. Moreover, certain tools are unable to monitor QoS attributes and SLA requirements at the application level (e.g., security, elasticity, performance) because they mostly focus on monitoring hardware resources (CPU, storage, and networks). Besides, most commercial tools (e.g., CloudWatch, LogicMonitor) are not sufficiently flexible to allow a service provider to extend the QoS attributes provided to be monitored to assess SLAs' fulfillment.

In summary, despite the number of approaches proposed to monitor the quality of cloud services, there is still a need for flexible solutions that allow monitoring parameters to be added or modified at runtime. These requirements may occur due to the need to measure values of specific quality attributes that were not of interest when the conditions were initially established. To address this problem, we proposed in previous works a monitoring approach that uses models at runtime to monitor quality requirements of cloud services. In addition, a monitoring infrastructure was then proposed to support this approach. This infrastructure integrates two main components, a monitoring configurator and an analysis and monitoring middleware which are briefly introduced in Section 3.

Although our preliminary results support the hypothesis that MoS@RT can be considered as a promising approach to monitor the quality of cloud services, further evaluations in other domains are needed. This is exactly the objective of this paper which intends to study the feasibility of applying MoS@RT to an AAL environment.

2.2. Ambient Assisted Living Challenges

One of the most critical challenges worldwide is the efficient delivery of healthcare services for the aging population (National Academies Press 2001). Personal care and older people's relatives are concerned if they are alone in their homes and several circumstances happen that affect their well being. Thus, enabling technologies for independent living such as AAL systems can be an innovative way to enhance care cost-effectively. However, these systems are critical since, in case of failure, they may cause damage to human lives. Therefore, the assurance of quality requirements is considered a key concern during the development of this kind of systems.

The Internet of Things devices can be included in these systems to collect relevant information from users (e.g., their body) or environment (e.g., home, public spaces). Those devices are highly heterogeneous and need to work together to provide an effective solution (Vora et al. 2019; Yoo et al. 2019). The data provided by those devices are stored in the cloud; therefore, several data sources need to be considered. Information supplied by monitoring solutions need to be provided in order to measure the service quality, and there are several studies that address this area. For example, (Cristescu et al. 2020) presented a set of specific quality measures to evaluate the quality in use of an AAL platform that monitors and supports elderly. These authors emphasize the importance of specifying and evaluating the relevant quality characteristics using validated and widely accepted metrics.

On the other hand, the cloud, which interacts with the IoT devices, needs to be monitored to assess the quality of cloud services. It is essential because, with monitoring information, it is possible to trigger alarms or take corrective actions concerning the patients' health. In this sense, several Home Monitoring Systems (HMS) frameworks have emerged as an alternative to classic healthcare services for the aging world population (Achirei et al. 2020). However, most of the solutions are not centered on cloud monitoring. Instead, they consider its particular characteristics and are mainly concerned with the capture of data from the patients and the environment (Stavrotheodoros et al. 2018).

2.3. Emerging domains in cloud environments

Currently, emergent technologies have been leveraging new challenges related to the quality of systems that manage or control them. Thus, there is a need to provide services that accomplish the expectations of end-users. For example, wearables and sensor nodes are integrated into IoT systems to monitor patients, provide security and health services, and provide high-quality demand systems. Those systems transmit data to cloud servers via the gateway in real-time. In addition, the cloud platform acts as a bridge between sensors and web browsers (Yang et al. 2018).

There are several solutions where cloud computing acts as part of the entire system; thus, it is essential to maintain a high quality of the provided services; in this context, monitoring is the first step toward that quality of service. Therefore, the health is a domain in which cloud platforms work together with devices. Here, (Yang et al. 2018) present a wearable device with a bio sensing facial; this solution acts as a wireless sensor node, which is integrated into an IoT system for remote pain monitoring. In the sensor node, up to eight channels of sEMG can be sampled; therefore, data are transmitted to the cloud server via the gateway in real-time. This solution needs scalability and performance quality characteristics to receive, store and process data.

Another example is the solution provided by (Forooghifar et al. 2019), where authors aim to distribute the complex and energy-consuming machine-learning computations between the edge, fog, and cloud, based on the idea of self-awareness, which needs to realize the system health. Another study that evidence the need for monitoring systems related to cloud computing is presented by (Wang et al. 2019), which proposes a framework for environmental monitoring based on fog computing that uses multi-source heterogeneous data gathered from IoT sensors. They analyze the collected data and design a federated learning mechanism for their solution. Finally, they evaluate the framework by employing an experiment.

Consequently, the quality of services is a concern to accomplishing those solutions' objectives. However, it is necessary to include a variety of data sources. Therefore, a flexible monitoring mechanism is needed to support and unify all data sources, providing a unique solution that integrates any device or monitoring specialized solution.

3. Monitoring Infrastructure

Cloud MoS@RT is a technological service infrastructure developed to support the MoS@RT method for monitoring cloud services. The main benefit of Cloud MoS@RT is that it provides a flexible and high-level monitoring service solution that is independent of cloud providers. This service is based on models at runtime and allows changing at runtime the monitoring requirements, or the metric operationalizations of these requirements, without the modification of the underlying service infrastructure. In addition, Cloud MoS@RT has been empirically evaluated through controlled experiments to study the users' perceptions of ease of use, usefulness, and future use intention with promising results (Cedillo et al. 2015, 2016).

Figure 1 shows an overall view of the Cloud MoS@RT functionality, including their components, activities, and models employed. There are three main components: Planner, Configurator, and Monitor.

The **Planner** is in charge of *Establish the Monitoring Requirements*. This activity is performed to collect those requirements to be monitored, which come from the SLA and additional quality requirements of interest, together with their corresponding thresholds. This information is stored in a Monitoring Requirements Model.

The **Configurator** uses the Monitoring Requirements Model and a SaaS Quality Model (i.e., an ISO/IEC 25010-compliant quality model for cloud services (Guerron et al. 2020)) to set up the monitorization. The SaaS Quality Model contains SaaS quality characteristics, attributes, and possibly several metric operationalization alternatives that can be used for monitoring the cloud services.

The activities performed in the *Configurator* are:

i) Select the Quality Attributes, consisting on the selection of the quality attributes from the SaaS Quality Model associated to each monitoring requirements;

ii) Select the Metrics, consisting on the selection of the concrete metric operationalizations to be applied in order to assess the quality of the service;

iii) Map the Metrics to Data Sources, consisting on the mapping between the metric definition and the corresponding data source for the metric in a given platform (e.g., in Microsoft Azure the data source will be the Azure Performance Counters, which can be obtained by using the Microsoft Azure Diagnostics Service) and,

iv) Runtime Quality Model Generation, which generates the model at runtime. This model at runtime contains the structures that reflect the set of specific quality requirements to be monitored in terms of the Saas Quality Model, and it is causally connected to the cloud service to be monitored, meaning that a change in the runtime model will trigger a corresponding change in the data source that is gathered for monitoring and/or vice versa. This facility allows us to easily change at any time the monitoring requirements model, the SaaS quality model, or any aspect of the data sources for the metrics in the cloud platforms.

Finally, the **Monitor** is composed of a Measurement Engine and an Analysis Engine. The Measurement Engine, which is implemented as a service to interact with the cloud service to be monitored, uses the Runtime Quality Model to gather the raw data from the cloud service an is responsible to *Perform the Measurements*. Then, the Analysis Engine uses the Runtime Quality Model to *Analyze the Results* obtained by determining if the data gathered from the monitored service violate the defined thresholds for the metrics associated to the quality requirements.

3.1. The Monitoring Requirements Metamodel

This metamodel defines the structure of the Monitoring Requirements Model for the quality requirements to be monitored. It is compliant with the WSLA Language Specification (Keller & Ludwig 2003) to allow representing these quality requirements in an standardized manner and to facilitate their evaluation. Moreover, it is possible to include in this model not only quality

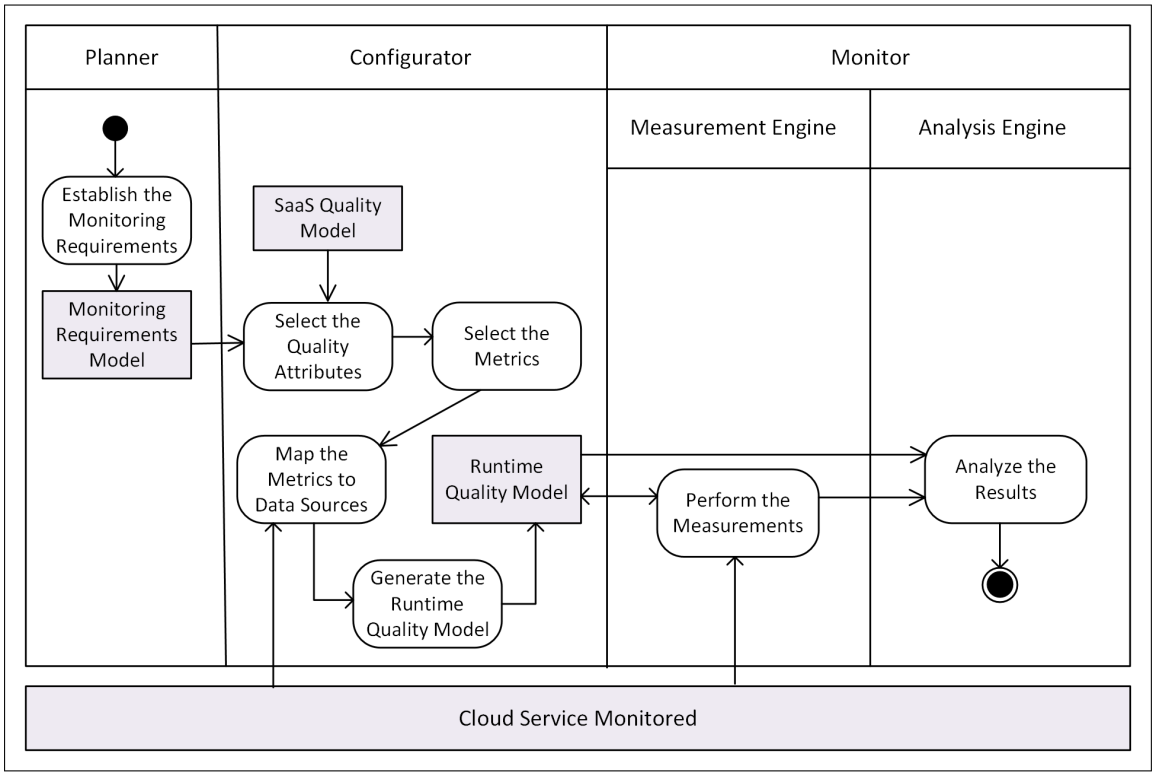

Figure 1 Cloud MoS@RT Monitoring Activities

requirements from the SLA of the services to be monitored but also additional requirements that may be of interest to stakeholders.

Figure 2 shows the monitoring requirements metamodel, which includes the main SLA sections. The *SLA Document* specifies the parties, service definition and the obligations. The *parties* are the signatory parties, namely the service provider and the service customer, and the supporting parties that provide the service measurements and audits. The *Service Definition* is the abstraction of the service to be monitored, whose quality characteristics and attributes are relevant as regards defining the SLA's terms. Characteristics and attributes are specified as SLAParameters that can be measured by using Metrics.

Finally, the *Obligation* metaclass contains two types of obligations: i) the *Service Level Objective*, which is a warranty of a particular state of SLA parameters in a given time period (e.g. the *average response time* must be 5 ms); and ii) The *Action Guarantee*, which specifies the provider's commitment to doing something in a specific situation (e.g. if a violation of a guarantee occurs, a notification is sent specifying a penalty). A more detailed explanation of the use of WSLA for specifying requirements can be found in (Keller & Ludwig 2003).

3.2. The SaaS Quality Metamodel

Figure 3 shows the SaaS Quality metamodel, which is aligned with the ISO/IEC 25010 standard (ISO/IEC 25010 2011). It allows the definition of *characteristics*, *sub-characteristics*, *attributes*, *impact*, and *metrics* that specify how the quality requirements of cloud services can be measured. Also, it is possible to define a set of *operationalization* alternatives for each metric. These alternatives are useful since several quality requirements (e.g., scalability, elasticity, security) need to be monitored at different cloud service levels (i.e., SaaS, PaaS, IaaS) (Aceto et al. 2013). The *Direct Metric* metaclass is used to represent metrics that do not depend on any other metric and whose values can be obtained directly from the raw counters of the cloud platform, whereas the *Indirect Metric* metaclass is used to represent metrics that are derived from others metrics and that need a *Measurement Function* to calculate their corresponding values.

Finally, the *Platform* and the *Measurement Method* and *Measurements Function* are used to to maintain a list of raw platform-dependent data counters and the platform-specific algorithms, respectively. The *Unit* metaclass includes the magnitude related to a particular quantity, and the *Scale* metaclass represents a set of values with continuous or discrete properties used to map the operationalizations.

3.3. The Runtime Quality Metamodel

This metamodel defines the structure of the Runtime Quality Model (model at runtime). This model specifies the monitoring requirements, metrics, operationalizations, and configurations that will be used during the monitoring of the services.

In (Lehmann et al. 2010) the following modeling constructs for a model at runtime are defined: (a) a *prescriptive part* of the model, which specifies what the system should be like; (b) a descriptive part of the model, which shows what the system is like; (c) the *valid model modifications* of the descriptive parts, which are formulas or sentences that can be executable

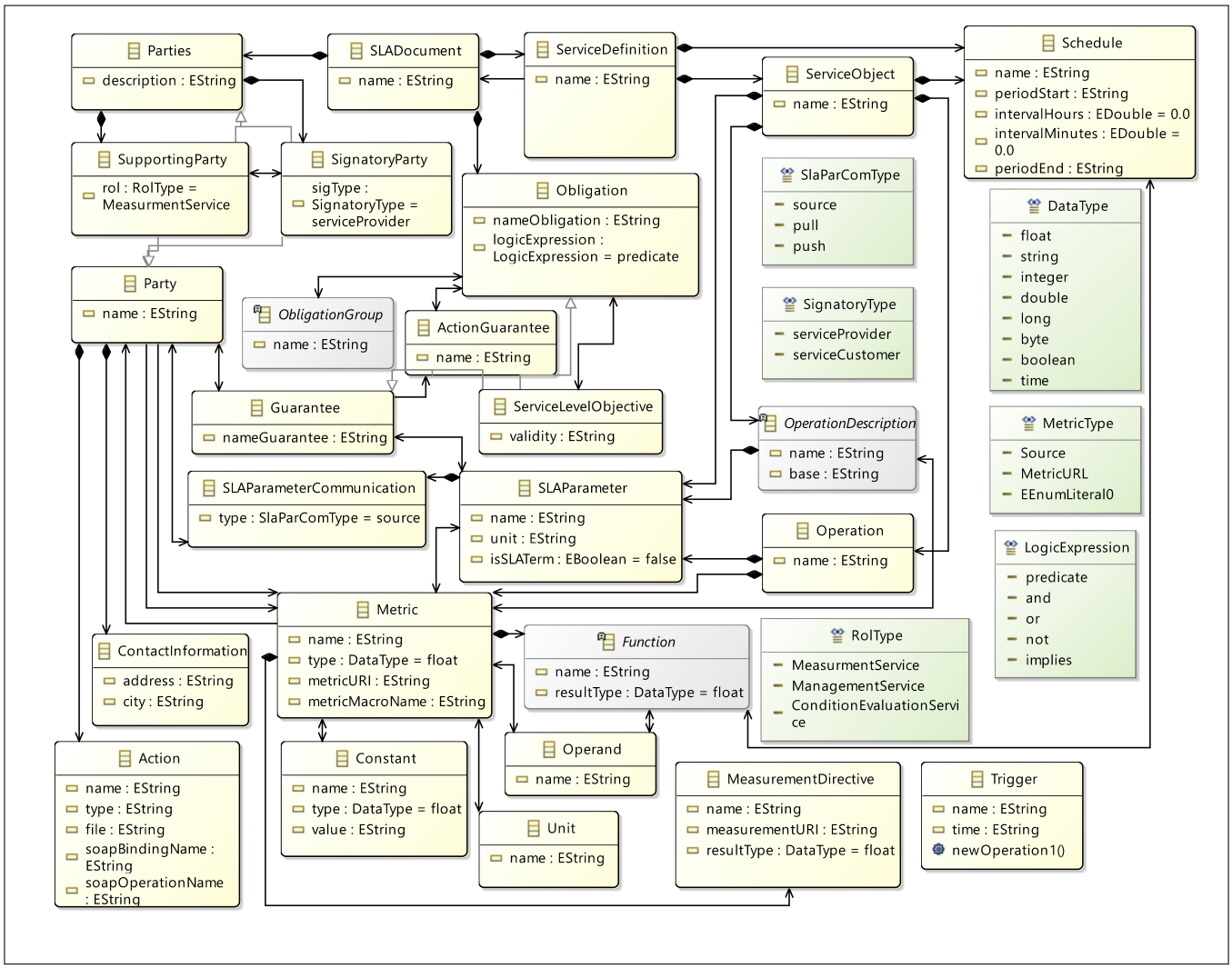

Figure 2 Monitoring Requirements Metamodel

at runtime; (d) the *valid model modifications* of the prescriptive parts, which are formulas or sentences that can be executable at runtime; and finally (e) the *causal connection*, which represents the information flow between the model and the entity to be observed (monitored).

Figure 4 shows the Runtime Quality Metamodel, which is based on the SaaS Quality Metamodel and that also incorporates the following parts: i) a *prescriptive part* to represent the quality requirements and relevant SLA terms with the corresponding thresholds; ii) a *descriptive part* to represent the specific raw data and calculated metric values resulting from the measures; and iii) a *causal connection part* to represent the specific characteristics of the connection to the cloud platform allowing the flow of information between the model and the data obtained from the monitored service. The *model modifications* (prescriptive and descriptive) are represented as the *measurement functions* that calculate the corresponding metrics, and the *mappings* that associate the raw data data source in a given cloud platform to the metrics.

Specifically, the *Cloud Service* metaclass describes the service to be monitored. The *prescriptive part* includes the *Thresholds* obtained from the SLAs for the specific metrics associated to the quality requirements. The *descriptive part* includes the *RawData Instance*, which contains the values captured directly from the cloud service, and the *Calculated Metric*, which contains the measurement results of the calculated metrics. The *Configuration File* contains specific information for each platform that allows the interaction between the monitoring infrastructure and the cloud service. This interaction between the model and the monitored service is a causal connection that allows not only to continuously observe and interpret the row data obtained from the monitored service but also to change this interaction when the model changes (adding a new quality requirement, changing a metric or the operationalization of a metric, etc.).

4. Applying the Monitoring Infrastructure

An exploratory case study was performed by following the guidelines presented in (Runeson et al. 2012) in order to study the feasibility of applying MoS@RT to monitor the quality

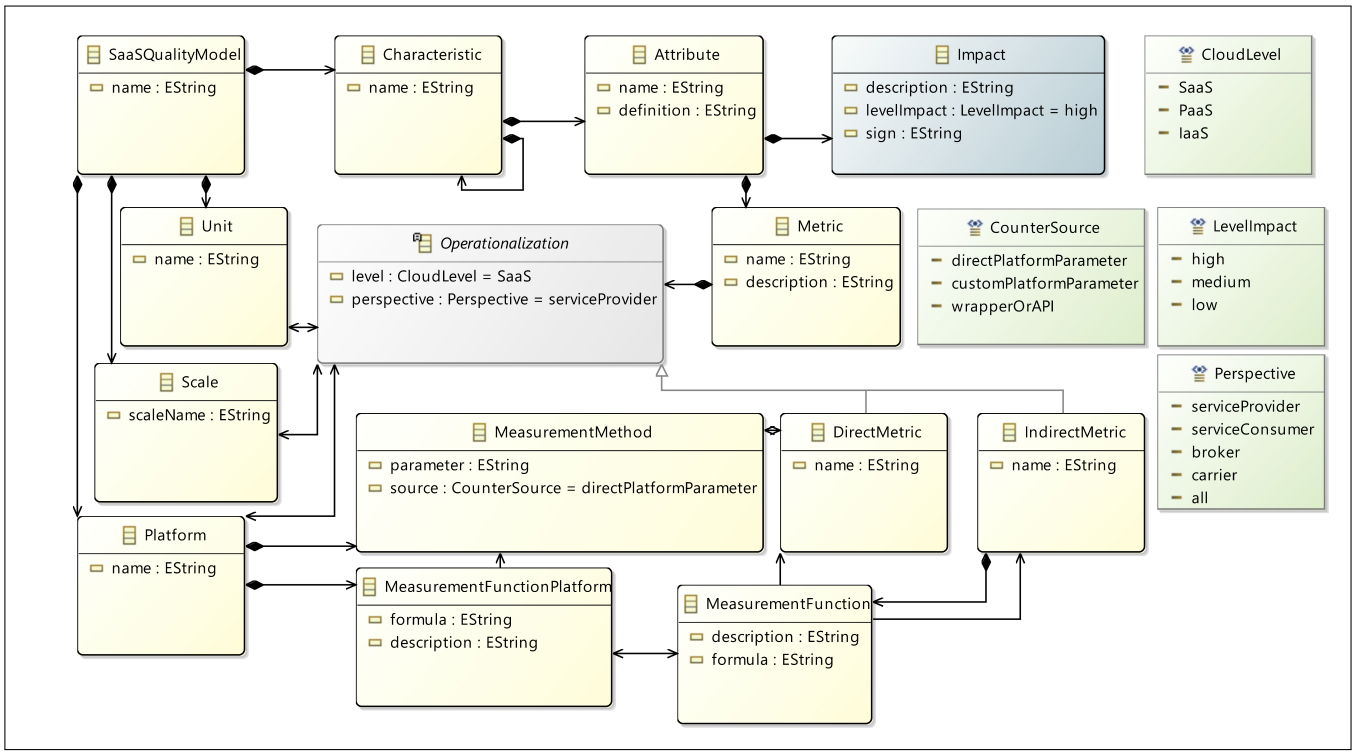

Figure 3 SaaS Quality Metamodel

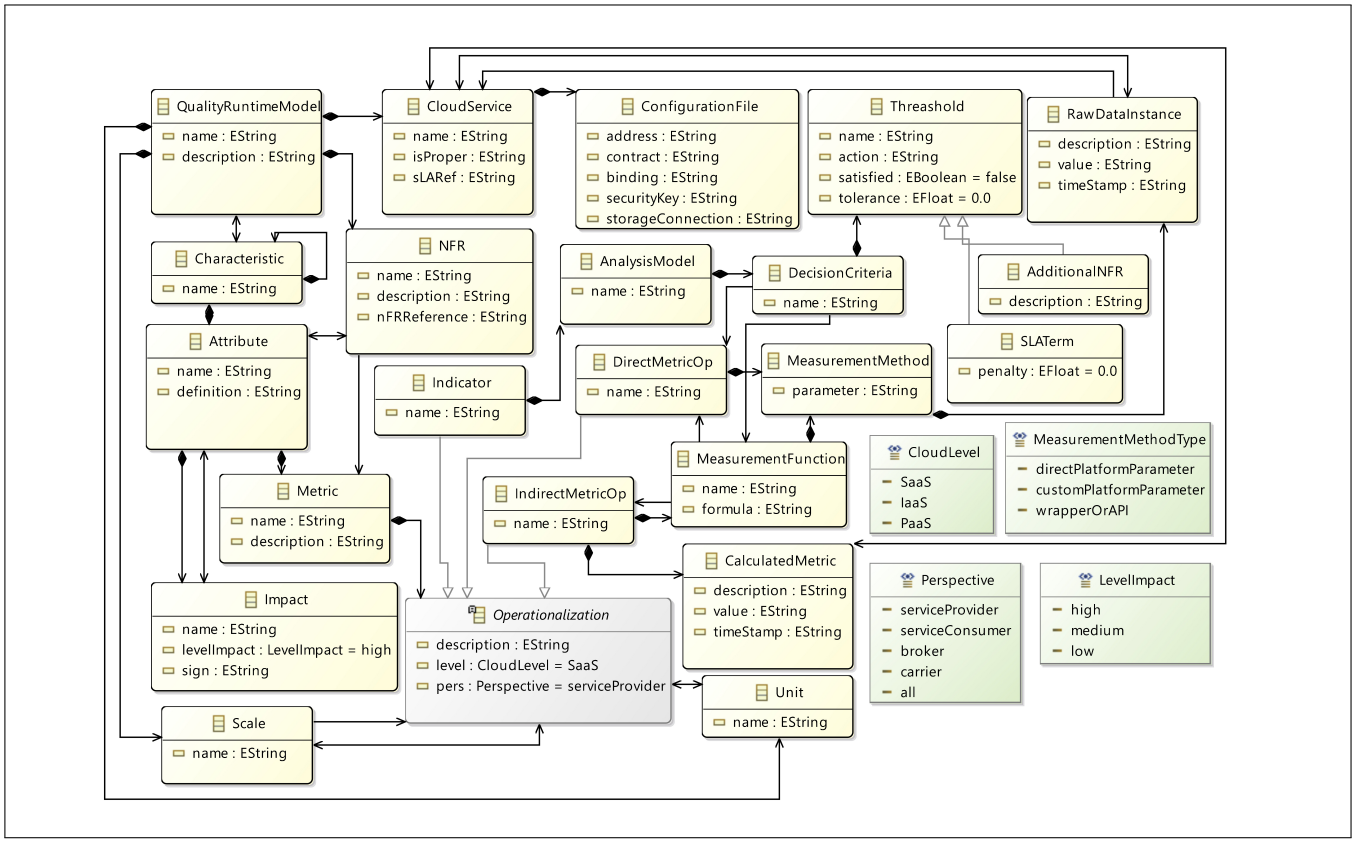

Figure 4 Runtime Quality Metamodel

of cloud services in an ALL environment. A case study is an empirical inquiry that investigates a phenomena in real-life contexts. The stages of which the case study was comprised were: design, preparation, collection of data, and analysis of data, each of which is explained below.

4.1. Design of the Case Study

The case study was designed by considering the five components that are proposed in (Runeson et al. 2012): purpose of the study, underlying conceptual framework, research question to be addressed, sampling strategy, and methods employed.

The purpose of the case study is to show the feasibility of applying the MoS@RT method, its infrastructure (Cedillo et al. 2016), and middleware (Cedillo et al. 2015) to monitor services deployed in the cloud corresponding to an AAL environment.

The conceptual framework that links the phenomena to be studied is the configuration and quality assessment of the deployed services in the AAL environment according to the steps established in MoS@RT. The AAL environment was selected considering its heterogeneity and service-oriented characteristics. In addition, in these environments, cloud computing infrastructures are regarded as a critical solution for processing and storing data provided by IoT devices. First, the monitoring requirements were established. Then, the monitoring configuration has been carried out to generate the runtime quality model. Next, raw data were collected from the services distributed in the environment to perform the measurements. Finally, the quality assessment were performed according to the runtime quality model. It is worth noting that the quality assessment process performed by our approach is compliant to the ISO/IEC 25040 standard (ISO/IEC 25040 2011).

4.1.1. Research question. We are interested to determine whether Cloud MoS@RT is able to monitor a set of quality attributes that are relevant for the AAL domain. Therefore, the research question is stated as follows:

– RQ: Can relevant quality attributes of AAL systems be properly specified by means of Cloud MoS@RT Configurator?

4.1.2. The case and the unit of analysis. The sampling strategy of the case study is based on an embedded single-case design. We contacted a development company located in Cuenca (Ecuador) in order to apply Cloud MoS@RT to a real AAL environment from a project in progress, as shown in Figure 5. The company provided us with access to a system that exploits cloud services and IoT devices.

This system aims to provide elderly with the possibility of improving their autonomy and safety through devices hosted in their homes. Thus, the AAL solution includes IoT devices that meet these objectives. This AAL system has been selected because it needs to warranty high-quality services due to the importance of the patient health and well-being. The cloud is responsible for managing and process data collected from different IoT edge sensors. Consequently, the need to monitor aspects provided with technology that is responsible for, through sensors and actuators, providing the environment that the person needs have been determined. Furthermore, these systems must have an analysis and assurance of the quality of the services that manage the devices located in the AAL environment.

In addition, it must be considered that IoT devices are required within the environment. In these environments exist a broad type of sensors and actuators located in the house, and some of them are wearable used by the elderly. Still, there must be information storage and data processing mechanisms in addition to them. Therefore, there is a front-end part within the system architecture deployed in a community cloud provided by CEDIA (Ecuadorian Corporation for the Development of Research and Academia). The back-end and the database were deployed in Heroku (Salesforce Company 2020), as a private cloud. This cloud application platform must be continuously monitored to ensure that data and processes are safe and verify if the SLAs are adequately met. This cloud platform allows the provisioning of applications and data storage, and it also provides specific metrics for monitoring cloud services.

Figure 5 shows the distribution of the sensors, actuators, and other elements that send and receive data and actions to and from the cloud. In particular, the AAL environment has three places with different needs and priorities regarding the management and use of information: i) living room, ii) kitchen, and iii) bedroom. The system has three different layers depending on the deployment infrastructure: edge, fog, and cloud.

At the edge computing layer, edge nodes (i.e., IoT devices distributed in the environment) represent gateways and data capturing services able to act on raw data, for example, aggregating, filtering, and encoding the local data streams in real time. Specifically, the IoT platform uses a temperature sensor in the living room and a wearable with a heartbeat sensor located in the bedroom and a carbon monoxide (CO) sensor in the kitchen. This layer is where the cloud resources are being distributed and moved near the end-users and end devices.

These sensors periodically collect data from home and send them to the cloud or a fog computing node.

At the fog computing layer, the monitoring solution has been installed and deployed in a private cloud located in the user's home. In this layer, data are processed and stored from the sensors to reduce latency, network traffic, and timely control of devices located at the edge of the IoT infrastructure. The sensors and the server communicate asynchronously using a wireless network (WiFi) with the support of the Message Queue Telemetry Transport (MQTT) protocol.

Finally, at the cloud layer, there is a computing node for the deployment of the AAL system presentation layer, which has been deployed on the CEDIA cloud (*CEDIA Web Page* 2022). It stores the data resulting from the aggregation operations executed in the fog node. It also includes a public network where certain aspects (e.g., reports with summarized data from the fog node, critical information and certain massive data that require immediacy due to its criticality) will be stored and processed.

In summary, the AAL environment contains a set of IoT devices, which are made up of a series of sensors and actuators. These devices use web services, store the generated data in the Heroku public cloud, and monitor the environment through the cloud MoS@RT infrastructure which was deployed in a privated

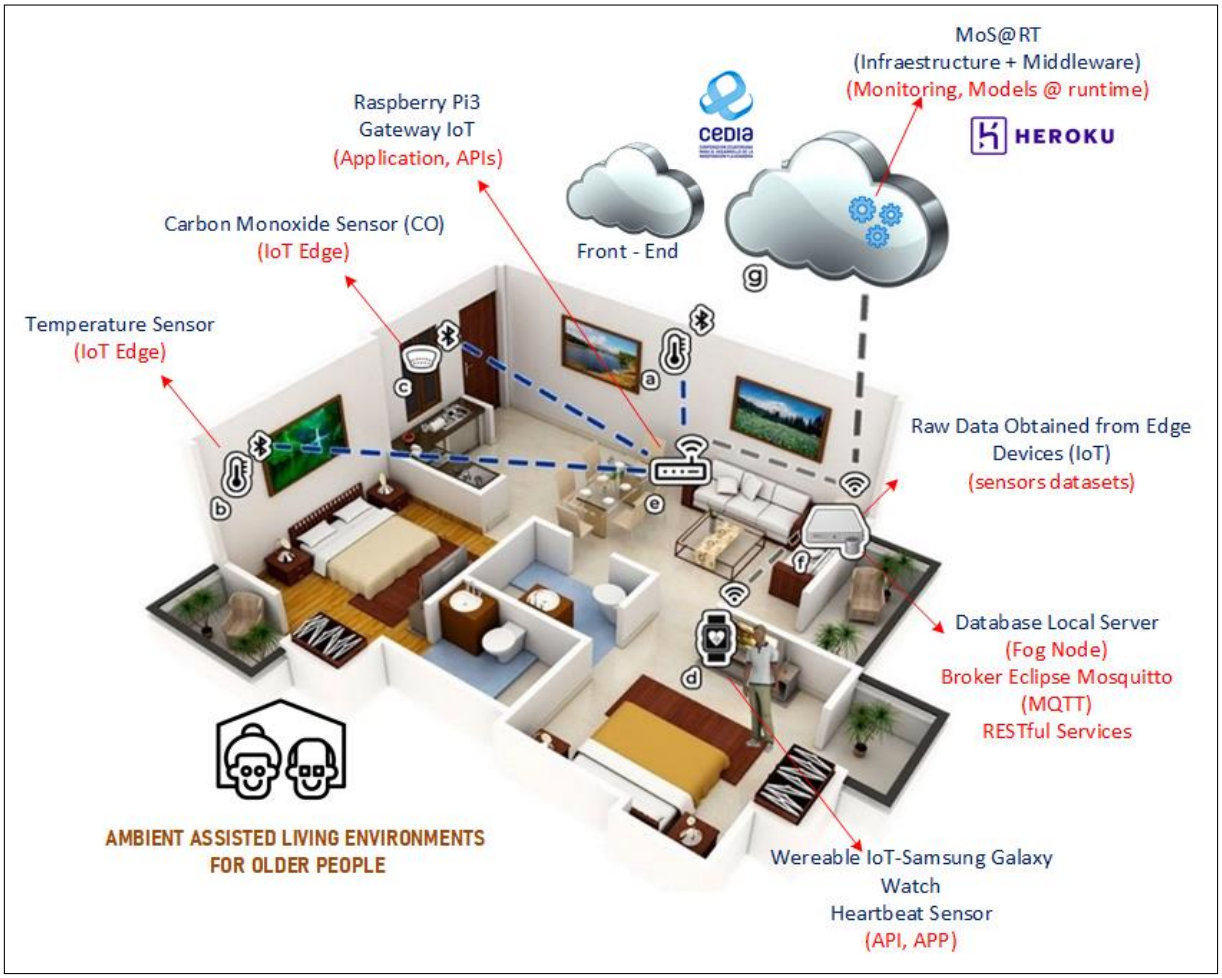

Figure 5 AAL system infrastructure

cloud as a service monitoring system.

The **unit of analysis** are the three services deployed in the cloud (i.e., a service to manage the temperature sensor, a service to manage the carbon monoxide sensor, and the service to manage a heartbeat sensor included into a wearable) as shown in Figure 5 and the monitoring of the data gathered by them.

4.2. Preparation of the Case Study

Cloud MoS@RT was applied as follows: two domain experts played the role of *Monitoring Planner* by providing us with the monitoring requirements; then, the main author played the role of *Monitoring Configurator* by selecting the appropriate quality attributes and metrics to be monitored, establishing a mapping between the metrics to specify how each metric will be calculated in the Heroku platform, and generating the runtime quality model. This model was then used by the monitoring infrastructure to collect data in real-time from the services in operation, apply the metrics and provide the monitoring data. Finally, the infrastructure performed the data analysis and presented the monitoring results. These steps are further explained in the following subsections.

4.2.1. Establishing the monitoring requirements. The aim of this stage is to carefully examine the domain of interest, establish the purpose of the evaluation, establish the requirements and delimit the scope of monitoring.

The *goal* of this evaluation is to assess the overall quality of the AAL system (see Section 5) and decide on the acceptance of the product. Two domain experts (i.e., a geriatrician and a psychologist) and three software developers involved in the development of the AAL system have been asked to propose the relevant quality requirements for the system.

The requirements were defined during a workshop where we discussed with these stakeholders the objectives of the AAL environment, the information needed, and the cloud infrastructure characteristics and behavior. The SLA was defined by considering the provided non-functional requirements. Additionally, the SLA was specified for the three services to be monitored; consequently, the stakeholders selected those clauses that were of interest to them. We noticed that the defined quality requirements are mainly related to the immediacy and criticality of the information managed by the system (i.e., the heartbeat, carbon monoxide, and temperature sensors to be monitored are critical for patient health and security), and the use of resources. Specifically, the requirements to be monitored are the following:

1. The capacity and memory use of the virtual or physical machine in which the service is executed should have a minimum of 512 MB.

2. The memory capacity used by an active service before load balancing must be a maximum of 75% of the total assigned memory to guarantee the proper functioning of the service.

3. The memory usage of a virtual machine before load balancing should be 80% to ensure client resources.

4. The delay in provisioning the service will be a maximum of 550ms.

5. The service delay time and the resource request time should have a maximum of 500 ms.

6. The service availability regarding access to the network must be 24/7 with a maintenance time of 6 hours maximum per year.

7. The latency of a user request must be a maximum of 50 ms.

8. Response time should not be more than 8000 ms.

9. The number of requests adequately served per second is 15 requests.

10. The incidence of accidents must be a maximum of 0.05 per 100 requests.

11. The number of correct operations must be at least 99.5%.

12. The memory capacity will be a minimum of 512 MB for each virtual machine.

13. The negotiated time for service availability is 24/7, excluding the maintenance time of 6 hours maximum per year. Therefore, the unavailability of the service cannot be more significant than 0.001%.

14. In the event of any invalid connection attempt, a message from the source IP will be sent to the administrator's email to proactively identify the dangers to the security of the cloud service.

15. In addition, every time more than 500 requests are made per minute, an alarm will be triggered to prevent denial of service attacks.

16. A maximum loss of stored data corresponding to 0.000001% per month must be guaranteed.

17. A maximum of 10 requests for minor changes in the SaaS will be allowed once the service has been deployed. These changes will be free of charge and will not represent a difficulty that impacts the database. Additional changes will be billed according to the difficulty in the implementation and redeployment.

18. None of the contracted services will require payments or the need for third-party operations, but only those agreed with the customer, in this case, Google Maps, the only interdependent service for geo-referencing patients.

In this AAL environment, quality attributes such as performance, response time, usage of resources, and reliability are essential to provide users with services that make good use of resources and perform the specified functions under the stated conditions as expected. Also, it is worth mentioning that we focus on measuring internal and external quality attributes that are currently supported by MoS@RT. During the workshop carried out with the stakeholders, other quality attributes related to the quality in use of the services (e.g., user experience, satisfaction) were also of interest to the company, but these were discarded because this type of attribute are not yet supported by our monitoring approach. Moreover, in a home, it is challenging to include devices such as cameras, which can represent privacy-invasive devices for users. After all, users might not feel so comfortable with these devices.

These requirements have been specified in a Monitoring Requirements Model which conforms to the monitoring requirements metamodel shown in Figure 2. This model contains the detailed specification of each one of the requirements including the thresholds from the SLA clauses. Figure 7 shows an excerpt of the Monitoring Requirements Model for the AAL system.

4.2.2. Specifying the evaluation The aim of this stage is to design how the monitoring will be performed and what information will be collected during the services monitoring. Specifically, the activities to be carried out correspond to those of the MoS@RT Configurator. These activities are aligned with the ISO standard (ISO/IEC 25040 2011) for the evaluation of software product quality.

The first activity is to *select the quality attributes* to be monitored. We selected a set of attributes from the SaaS Quality Model which conforms to the SaaS quality metamodel shown in Figure 3. This model is a subset of the Cloud Services Quality Model proposed by (Guerron et al. 2020) which represents the decomposition of cloud services quality characteristics into measurable quality attributes.

The selection of quality attributes has been performed to cover the monitoring requirements established as part of the workshop conducted with the system stakeholders. Table 1 shows the selected quality attributes.

The next activity is to *select the metrics* that will allow to evaluate the selected quality attributes. The SaaS Quality Model contains different metric operationalization alternatives for certain quality attributes that can be used during the service monitoring process. The operationalization of a metric consists of establishing a mapping between the generic specification of the metric and the concepts that are represented in the software artifacts to be measured. In this way, a quality attribute can be measured using different measurement functions depending on which artifact (e.g., cloud service specification, cloud architecture, actual cloud service) is being assessed. The possibility of having several operationalizations also allows the most appropriate measurement function to be selected (e.g., by considering

Quality Attribute	Definition
Memory capacity	Represents the memory size of the cloud service
VM memory usage	Represents the percentage of the memory resource occupied.
Delay	Represents the delay that can be introduced by the network transport.
Internet access	Represents the ability to attend to requests made to the service in response to a workload
Latency	Refers to the time necessary for a user's request to be dealt with by the competent service.
Response time	Represents the response time for a request to be fulfilled or the amount of data generated by the cloud service in a specific time interval. Synchronous and asynchronous treatment should be considered.
Performance	Represents the volume of requests per unit of time. A request can be a job, task, operation. (WIPS as web interactions per second, OPS as operations per second, TPS as transactions per second).
Accident incidence	Represents the relationship between the number of accidents in the service and the number of accidents allowed in the SLA.
Faulty operation	Represents the ability of an item to perform a required function under stated conditions for a stated period.
Availability time	Represents the availability time to access the service, and customers can use it.
Operational availability	Measures a ratio between total up time and the time the SaaS is available to be invoked.
Security risks	Represents the ability to identify cloud service security threats proactively.
Monitored alarms	Represents the effectiveness of the alarms monitored by Corporate Security.
Data integrity	Represents the ability to verify that the service stores data correctly. Data integrity is a broad term and includes data security, privacy, and accuracy.
Flexibility	Represents the ability of the service provider to regulate changes in services according to customer requests. Is the ability to adjust or change the cloud service in response to customer needs or changes in technology.
Service modularity	Indicates if the service is self-contained, its goal is to minimize the components' external dependencies.
Interoperability with dependent services	Represents the level of efficient interactions between the service and its subordinate services in all business processes in which the service participates.
Composability	Indicates that services by incorporating other services are more easily and effectively customized to meet the specific needs of users.

Table 1 Selected Quality Attributes from the Cloud Services Quality Model

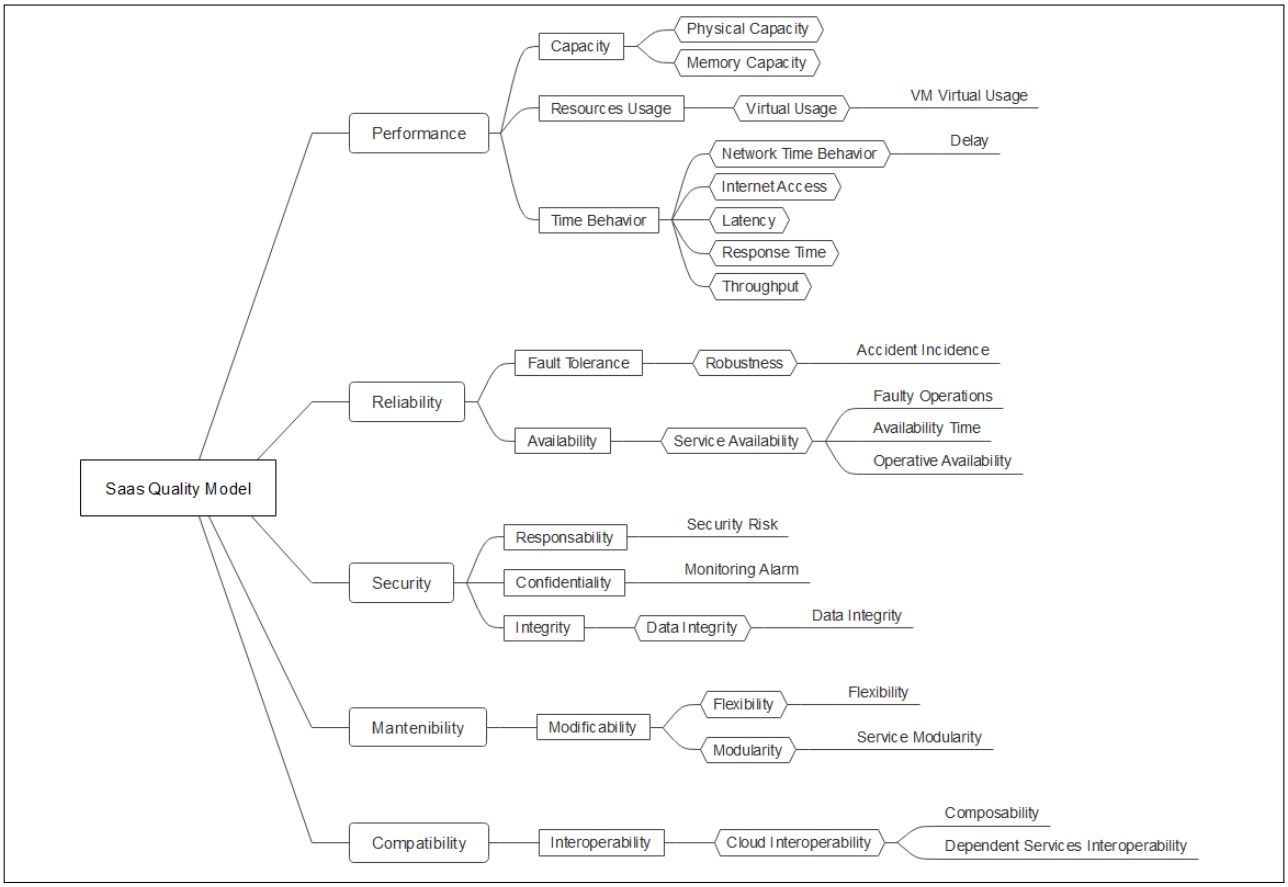

Figure 6 A SaaS Quality Model (Guerron et al. 2020)

the availability of raw data in a specific platform).

Table 2 shows the resulting quality attributes, metrics and operationalizations that were selected from the SaaS Quality Model. Additionally, Figure 6 shows the decomposition of quality characteristics on subcaracteristics, quality attributes and metrics for the AAL system. This selection indicates that the quality characteristics most valued by the domain experts were Performance, Reliability, and Security while the quality characteristics most valued by the software developers were Maintainability and Compatibility.

4.2.3. Designing the evaluation In this stage the evaluation modules for the selected cloud platform and the decision criteria for quality measures are specified. The activities to be carried out correspond to the remaining activities of the Cloud MoS@RT Configurator: map the metrics to data sources and generate the Runtime Quality Model.

The first activity is to *map the metrics to data sources*. The objective is to define how the metrics expressed in the SaaS Quality Model (which are independent from cloud platforms) can be calculated in a specific cloud platform. This is done considering the actual parameters and instructions that are inherent to the selected cloud platform, which can be retrieved by using different methods (e.g., agents, APIs, platform tools, libraries).

Therefore, the SaaS Quality Model also contains the measurement functions that are used to calculate the metrics across different cloud platforms (i.e., platform-specific metrics).

As an example, the SaaS Quality Model contains three operationalizations (1) (2) (3) for the *Defective operations Per Million intents (DPM)* metric, as shown in Table 2. These operationalizations are independent of cloud platforms. Also, Figure 7 shows the specification of these metrics.

$$DPM = \frac{FailOperations}{AttemptOperations} \times 10^6 \quad (1)$$

$$DPM = \frac{AttemptOperations - SuccessOperations}{AttemptOperations} \times 10^6 \quad (2)$$

$$DPM = \frac{FailOperations}{(SuccessOperations + FailOperations)} \times 10^6 \quad (3)$$

In order to calculate the operationalizations in the Heroku platform for the DPM metric, which is an indirect metric, we needed to implement a wrapper that uses an API to gather the direct metrics associated to each IoT sensor. In particular, the direct metrics used in these operationalizations are: *AttemptedOperations*, *FailedOperations*, and *SuccesfulOperations*.

After defining the platform-specific operationalizations for each metric, we defined *decision criteria* to help evaluators to understand how to interpret the measurement results. Decision criteria are numerical thresholds or targets used to determine the need for action, or to describe the level of confidence in a given result. These criteria can be defined using benchmarks,

Quality Attribute	Metric	Operationalization
Memory capacity	Total memory size	Size of the service's memory resource, in megabytes (Mb), gigabytes (Gb)
VM memory usage	VM memory usage	Current occupied memory resource, in Mb, Gb
	Memory usage percentage	Current occupied memory resource / total memory resource * 100
Delay	Average delay time	Average value of the delay, in milliseconds(ms), nanoseconds(ns).
	Delay time	TR: time to obtain the resources – request time for the resources
Internet Access	Unanswered requests	AI = Number of unanswered requests / Total number of requests to the service. Percentage of unanswered requests.
Latency	Latency per request	Latency = average (request time-response time), in ms, ns
Response time	Maximum response time	It is the maximum response time of attention to a request in the cloud service in a specific time interval, in ms.
	Response time standard deviation (asynchronous task)	Standard deviation of the response time for an asynchronous task, in ms, ns
Performance	Service performance	The number of service requests handled / total service time
Accident incidence	Total accidents	Where, a=accidents number; b=accidents allowed in the SLA When a<b, IoA=1-a/b; When a>b, IoA=0
Defective operations	Defective operations Per Million intents (DPM)	DPM = (Fail Operations / Attempt Operations) * 10 ^6
		DPM = ((Attempt Operations – Success Operations) / Attempt) * 10^6
		DPM = (Fail Operations / (Success Operations + Fail Operations)) * 10 ^6.
Availability time	% Inactivity time	SD = (SF(t)/AG(t)) x 100% Where, SF (t) is the time of unavailability of the service during the negotiated time, AG (t) is the negotiated time.
	Uptime ratio	UT= ((total service time) - (total time during which the service was not available)) / (total service time)
		UT = Total Activity Time / Total Time
Operational availability	Service robustness	ROS = available time to invoke SaaS / total time to start operating SaaS.
Security risks	Number of safety hazards	Number of safety hazards proactively identified
Monitored alarms	Number of false alarms monitored	Number of false alarms monitored by Corporate Security
Data integrity	Resource data integrity	Check if the resource correctly stores the data. Rk (DI) = Dk / Ck Where, Ck denotes jobs successfully completed by resource Rk. Dk denotes the number of jobs that preserved data integrity by resource Rk during period T.
Flexibility	Ease of change	EoC = service updates / total time
Modularity	Service modularity	SM = Number of elements that do not depend on external services / Number of elements in a service
Interoperability with dependent services	Interoperability dependency	DI = Number of dependent services with successful interaction / Total number of dependent services in the participating processes
Composability	Service composability	Composability = W.SM + W.DI Where, SM Service modularity, DI interoperability dependency, W assigned weight based on importance, the sum of the weights must be equal to 1

Table 2 Quality attributes, metrics and operationalizations from the SaaS Quality Model

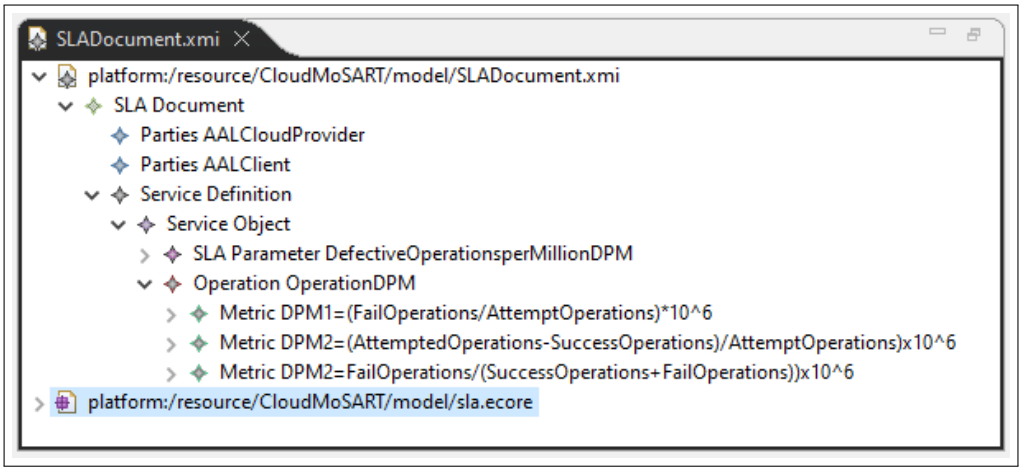

Figure 7 Monitoring Requirements Model Instance

historical data, customer requirements, etc. Since the decision criteria depend on the specific system domain in which the services are used, we defined these criteria with the stakeholders of the AAL system. The column *Threshold* of Table 3 shows the specified thresholds for the system under evaluation.

Once all these activities have been carried out, the MoS@RT Configurator allows the *generation of the Runtime Quality Model*. This is a model@runtime that specifies all of the monitoring requirements, metrics, measurement functions, and configurations that are needed to access the services to be monitored during their execution. This model corresponds to the *Evaluation Plan* described in the ISO/IEC 25040.

4.3. Collection of data

Monitoring data were collected from the cloud services of the Heroku cloud. This stage corresponds to the *execution of the evaluation* activity described in the (ISO/IEC 25040 2011).

In this stage, the Runtime Quality Model was used by the MoS@RT Monitor (Measurement Engine) *to perform the measurements* by gathering data from the cloud services of the AAL system and calculating the metrics, resulting in values on the measurement scales. The column *Result* of Table 3 shows the obtained value for each metric.

4.4. Analysis of Data

In this stage, the Runtime Quality Model was used by the MoS@RT Monitor (Analysis Engine) to *analyze the results* by applying the decision criteria to the value obtained for each metric obtained from the monitored services (i.e., service to manage the temperature sensor, service to manage the carbon monoxide sensor, and service to manage the heartbeat sensor). Thus, it was possible to determine that there are metrics that meet the established threshold and others that do not.

For example, Figure 8 shows the graphical representation of the measurement results performed during seven periods of time for some specific metrics. In *(a) VM memory usage*, we can see that the maximum and average values obtained are very close to the Memory capacity (512 Mb), exceeding 80% of this capacity in both cases. In *(b) Performance*, represents the number of attended requests in relation to the total number of requests. The average attended requests (reqs) was 9629 and the average unattended requests was 16 (calculated as the difference between the total number of requests and the attended requests), representing an average success rate of 99.83%. In addition, we have calculated the *(c) Delay time* for the three sensors interacting with its corresponding service. The fluctuation in time is shown in a time interval, where the minimum value is 4.94 ms, the maximum value is 128.12 ms, and the median is 12.16 ms. These values indicate that the requests could have different priorities to be attended or that server was overloaded by the number of requests. Finally, in *(d) Response time*, we have represented the 99th and 95th percentile, where it can be seen that the upper end is 117.75 ms and 92.15 ms, the median is 67.62 ms and 50.96 ms, and the lower end is 19.19 ms and 14.07 ms respectively, indicating that the largest number of requests has been attended to in the shortest time.

Finally, the column *Satisfies?* of Table 3 shows the analysis results. Those metrics that do not meet the threshold represent possible violations of the negotiated SLA or quality requirements that are not satisfied. Thus, the service provider may establish mechanisms for adjusting the service to meet the requirements, while the consumer may request the corresponding compensations established in the SLA clauses.

4.5. Answering the Research Question

The results show that all the monitoring requirements established for the AAL system could be monitored. The Cloud MoS@RT method and infrastructure provided the necessary guidelines and tools to select the appropriate quality attributes, metrics and operationalizations for this particular domain. We have instantiated our monitoring solution in the Heroku platform in order to be able to monitor any cloud service running in Heroku. Some existing data gathering mechanisms (wrappers) that we have previously defined for other cloud platforms (Azure and Google App Engine) could be adapted and used to this platform, but it was necessary to extend the Runtime Quality Model with new mechanisms (wrappers) to gather raw data to calculate some specific metrics in the Heroku platform. These

Metrics	Threshold	Result	Unit	Satisfies?
Total Memory Enabled	≥ 512	512	Mb	Yes
JVM Capability: Heap Memory Usage	≥ 512	512	Mb	Yes
VM Used Memory				
-Max Total	≤ 410	437	Mb	No
-Avg. Total	≤ 410	419	Mb	No
-Max Swap	0	0	Mb	Yes
-Max RSS	≤ 410	437	Mb	No
-Use JVM: Heap Memory	≤ 410	144	Mb	Yes
-Use JVM: No Heap Memory	≤ 410	136	Mb	Yes
Memory Usage Percentage				
-Max Total	≤ 80	85	%	No
-Avg. Total	≤ 80	82	%	No
-Max Swap	0	0	%	Yes
-JVM Use: Heap Memory	≤ 80	28	%	Yes
-JVM Use: No Heap Memory	≤ 80	27	%	Yes
Average Time Delay	$\leq 5,5x10^8$	Aprox $1x10^8$	ns	Yes
Delay Time	$\leq 5x10^8$	Aprox $6x10^7$	ns	Yes
Unanswered Requests	≤ 0.05	0.021	ms	Yes
Total Response Latency Per Request	≤ 500	8,000.54	ms	No
Response Time				
Maximum Response Time	$\leq 8,000$	13,311	ms	No
-99Th Perc	$\leq 8,000$	6,761.67	ms	Yes
-95TH Perc	$\leq 8,000$	5,096	ms	Yes
-50th Perc	$\leq 8,000$	748.33	ms	Yes
Response Time Standard Deviation height		38.09198	ms	
Accomplishment Over Time	$\leq 8x10^9$	8.37 x 10^9	ns	No
Defective Operations Per Million Attempts (DPM)	$\leq 5x10^-9$	$1{,}9x10^-8$		No
% Inactivity Time	≤ 0.001	0	%	Yes
Uptime Ratio	≥ 0.99999	1		Yes
Service Robustness	≥ 0.99999	1		Yes
Number of Safety Hazards	Informative	8		
Number of False Alarms Monitored	Informative	10		
Resource Data Integrity	≥ 0.999999	1		Yes
Ease of Change	Informative	1		
Service Modularity	≤ 0.80	0.67		Yes
Interoperability Dependency	1	1		Yes
Service Composability	≤ 0.90	0.83		Yes

Table 3 Monitoring Results.

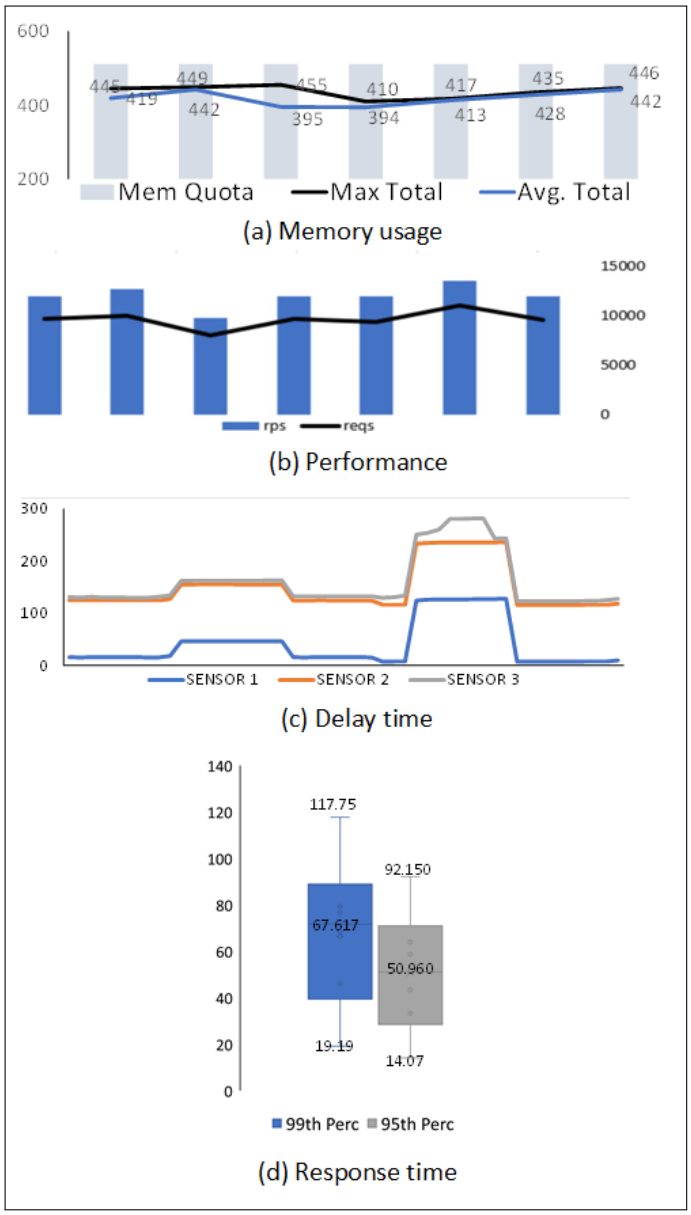

Figure 8 Selected Monitoring Results for the AAL system

wrappers support the gathering of customized information to calculate high-level metrics (e.g.,service robustness, resource data integrity, composability) that cannot be directly measured using the counters provided by the Heroku or other Heroku monitoring tools (e.g., Hosted Graphite Heroku monitoring).

There are diverse monitoring tools available that provide IT administrators with all the information they need to figure out whether there are any issues that are negatively impacting the quality of their service or not. However, most of these tools focus on the monitoring of performance by means of gathering low-level measurements by using performance counters provided by cloud platforms. Differently, our solution allow the monitoring of any quality characteristic because it is based on a quality model that decomposes these characteristics into sub-characteristics and quality attributes, which are then measured using existing metrics collected through a systematic literature review (Guerron et al. 2020). In addition, the application of models@runtime provides our solution with a high level of flexibility and maintainability because changes essentially affect the runtime quality model rather than the entire monitoring infrastructure.

4.6. Threats to Validity

Despite the fact that a systematic case study was carried out, it is possible that this work was affected by some threats to its validity. In this section, we shall, therefore, review the main threats and the actions taken to mitigate them.

4.6.1. Threats to Data Extraction and Analysis Results

With regard to the data extraction, we used well-known mechanisms to gather data and calculate the metrics (e.g., APIs, specific counters provided by the platform). These methods ensured a certain amount of consistency in the data gathering and analysis. However, it should be noted that in some cases, we had difficulties in extracting and interpreting the data owing to the fact that the information available about the selected metrics was not sufficiently clear or complete for us to be able to calculate the metrics. Note that these metrics were selected from an existing Quality Model for Cloud services that were defined by means of a systematic literature review (Guerron et al. 2020). The interpretation bias was, therefore, mitigated as far as possible by involving multiple researchers, having a unified scheme with which to gather the data and piloting the data extraction process with an external researcher. However, this study still has some limitations related to how data were obtained from different sources. We defined some wrappers to collect low-level information from the services that allowed us to collect data to calculate the selected metrics, but it is still required to extend the infrastructure with other mechanisms (e.g., plug-ins to existing cloud platform monitoring services, agents) to collect data for calculating other metrics of interest.

4.6.2. Threats to External Validity

This case study presents some limitations, such as the fact that only one AAL system was considered since the company imposed certain restrictions. Although the quality requirements to be evaluated were selected through the consensus of the five stakeholders involved in the system development and the evaluation designer, they might not have been very representative for all types of AAL systems. In order to overcome such limitation the study of (Garcés et al. 2017) can be used as a starting point in further studies since it determines which quality characteristics are critical for AAL systems and which ones are most relevant to some AAL sub-domains.

4.7. Lessons Learned

Our study reveals several findings that can serve as practical guidelines for both industry and academic communities to improve practices to monitor cloud services, in particular services used by AAL systems.

During the execution of this case study, several problems were identified. These problems occurred due to the time of deployment and configuration of the monitoring. First, it is

essential to consider that some metrics are difficult to measure. In this case, it is necessary to bear in mind that the data sources are varied and that it is necessary to select the most appropriate in each case. For this study, considering the established monitoring requirements, quality attributes and the metrics selected, it was necessary to create wrappers on many occasions, which requires additional development effort.

Second, companies require high-level indicators to help them make decisions about the tools that boost their profits. In this sense, monitoring services that retrieve raw data and low-level metrics are not enough. Our monitoring infrastructure bridge the gap between low-level measurements and high-level indicators. This is a crucial feature that was most valuable to the stakeholders in the company where this case study was conducted, and we believe that this solution represents a valuable tool for other companies.

On the other hand, many platforms bring closely linked solutions to them. However, it is essential to note that the broker or any actors who require SLA verification must know, deploy and maintain some monitoring solutions. Therefore, it implies that the management and maintenance of the cloud are very complicated, given its quality management and knowledge of SLA compliance. Thus, this type of cloud-agnostic tool they seek to monitor extensively facilitates work. It is also essential to know that companies' technological expansion and digital transformation have made cloud services highly demanded and used, so knowing how they work and their quality status is vital. Finally, emerging technologies work in highly heterogeneous environments, where data exchange is generally carried out through services. Results are stored in the cloud; this entails high number of transactions, requiring precision, security, and other essential quality characteristics to meet customer requirements. These technologies are related to the Internet of Things, intelligent environments, ambient assisted living, and smart cities. Therefore, the proposed monitoring infrastructure still needs to be evaluated in other domains.

5. Conclusions and Future Work

This paper has reported the results of a case study aimed at using a monitoring method and infrastructure (Cloud MoS@RT) to monitor the quality of cloud services in an AAL environment that has been deployed on the Heroku platform. We have instantiated our monitoring solution in this platform in order to be able to monitor any cloud service running in Heroku.

The goal of the case study was to assess the overall quality of a real-life AAL system aimed at increasing the quality of life of independently living elderly persons. The system runs in a heterogeneous environment that contains three layers (i.e., edge, fog and cloud). Two domain experts and three developers involved in the development of this system provided us with several quality requirements to be monitored.

The results suggest that relevant quality attributes of AAL systems can be adequately monitored using MoS@RT and that the report generated by the monitoring infrastructure is useful for service providers and customers to help them ensure that cloud services meet the required levels of quality. Those metrics that do not meet the threshold may represent possible violations of the negotiated SLA or quality requirements that are not satisfied. Therefore, the service provider may establish mechanisms for adjusting the service to meet the requirements, while the consumer may request the corresponding compensations established in the SLA clauses.

The use of models at runtime provides flexibility and eases maintainability because changes (e.g., adding a new quality requirement, changing the measurement function of a metric) essentially affect the runtime quality model rather than the entire infrastructure, as is the case with most existing monitoring solutions.

Current monitoring tools are focused on obtaining low-level data from the platforms and deployed services. Although this is necessary, this work goes one step further by creating a layer between the raw data and their interpreters (i.e., the decomposition of quality characteristics into subcharacteristics and quality attributes). This layer is useful as regard managing the huge amount of low-level data obtained from the platform and aggregating them to obtain high-level quality characteristics and attributes, which helps to make more informed decisions to improve the quality of services.

As future work, we plan to design and execute other case studies that involve the assessment of other quality characteristics and attributes relevant to the AAL domain. We also envision several ways to improve our monitoring solution.

Regarding the Configurator, we plan to improve its user interface to provide more guidance to navigate among the different models. For example, providing means to match the selected requirements from the Monitoring Requirements Model with the quality attributes and metrics from the SaaS Quality Model. In addition, since the cloud platform can provide a huge amount of raw data (and it is rarely well-documented), means are needed to help evaluators to select the appropriate parameters (e.g., performance counters) to define the platform-specific metric.

Regarding the monitoring infrastructure, we plan to study the self-adaptation capabilities of both the Runtime Quality Model and the cloud service being monitored due to their causal connection. When a quality requirement is not fulfilled, the cloud service could adapt itself instead of simply reporting the violation. Conversely, if the cloud service is changed, the Runtime Quality Model could identify the changes and adapt itself accordingly.

Finally, regarding the Analysis Engine, we plan to develop an interactive dashboard to support users to choose different mechanisms to visualize the monitoring results.

6. Acknowledgments

We thank to Fundación Carolina, Universidad de Cuenca, and Universitat Politècnica de València for their support. This research is supported by the project *Fog Computing applied to monitoring devices used in AAL environments: platform for the elderly* (Research Projects DIUC XVII).

References

Abrahão, S., & Insfran, E. (2017). Models@runtime for

monitoring cloud services in google app engine. *2017 IEEE World Congress on Services (SERVICES)*, 30-35. doi: 10.1109/SERVICES.2017.14

Aceto, G., Botta, A., de Donato, W., & Pescapè, A. (2013). Cloud monitoring: A survey. *Computer Networks*, *57*(9), 2093-2115. Retrieved from https://www.sciencedirect.com/science/article/pii/S1389128613001084 doi: https://doi.org/10.1016/j.comnet.2013.04.001

Achirei, S. D., Zvoristeanu, O., Alexandrescu, A., Botezatu, N. A., Stan, A., Rotariu, C., ... Caraiman, S. (2020). Smartcare: On the design of an iot based solution for assisted living. *2020 International Conference on e-Health and Bioengineering*, 1-4. doi: 10.1109/EHB50910.2020.9280185

Alhamazani, K., Ranjan, R., Jayaraman, P. P., Mitra, K., Liu, C., Rabhi, F., ... Wang, L. (2019). Cross-Layer Multi-Cloud Real-Time Application QoS Monitoring and Benchmarking As-a-Service Framework. *IEEE Transactions on Cloud Computing*, *7*(1), 48–61. doi: 10.1109/TCC.2015.2441715

Alhamazani, K., Ranjan, R., Mitra, K., Rabhi, F., Jayaraman, P. P., Khan, S. U., ... Bhatnagar, V. (2015). An overview of the commercial cloud monitoring tools: research dimensions, design issues, and state-of-the-art. *Computing*, *97*(4), 357–377. doi: 10.1007/s00607-014-0398-5

Baset, S. A. (2012, jul). Cloud slas: Present and future. *SIGOPS Oper. Syst. Rev.*, *46*(2), 57–66. Retrieved from https://doi.org/10.1145/2331576.2331586 doi: 10.1145/2331576.2331586

CEDIA Web Page. (2022). Retrieved 2022-02-10, from https://www.cedia.edu.ec/es/

Cedillo, P., Gonzalez-Huerta, J., Abrahao, S., & Insfrán, E. (2016). *A Monitoring Infrastructure for the Quality of Cloud Services.* Harbin, China. doi: 10.1007/978-3-319-30133-4_2

Cedillo, P., Insfran, E., Abrahão, S., & Vanderdonckt, J. (2021). Empirical evaluation of a method for monitoring cloud services based on models at runtime. *IEEE Access*, *9*, 55898-55919. doi: 10.1109/ACCESS.2021.3071417

Cedillo, P., Jimenez-Gomez, J., Abrahao, S., & Insfran, E. (2015). Towards a monitoring middleware for cloud services. *2015 IEEE International Conference on Services Computing*, 451-458. doi: 10.1109/SCC.2015.68

Cedillo, P., Sanchez, C., Campos, K., & Bermeo, A. (2018). A systematic literature review on devices and systems for ambient assisted living: Solutions and trends from different user perspectives. *2018 International Conference on eDemocracy eGovernment*, 59-66. doi: 10.1109/ICEDEG.2018.8372367

Cristescu, I., Balog, A., & Băjenaru, L. (2020). Quality in use measures for an aal system for older adults. *12th International Conference on Electronics, Computers and Artificial Intelligence*, 1-4. doi: 10.1109/ECAI50035.2020.9223192

Emeakaroha, V. C., Ferreto, T. C., Netto, M. A. S., Brandic, I., & De Rose, C. A. F. (2012). CASViD: Application Level Monitoring for SLA Violation Detection in Clouds. *36th Computer Software and Applications Conference*, 499–508. doi: 10.1109/COMPSAC.2012.68

Forooghifar, F., Aminifar, A., & Atienza, D. (2019). Resource-aware distributed epilepsy monitoring using self-awareness from edge to cloud. *IEEE Transactions on Biomedical Circuits and Systems*, *13*(6), 1338-1350. doi: 10.1109/TBCAS.2019.2951222

Garcés, L., Ampatzoglou, A., Avgeriou, P., & Nakagawa, E. Y. (2017). Quality attributes and quality models for ambient assisted living software systems: A systematic mapping. *Information and Software Technology*, *82*, 121-138. Retrieved from https://www.sciencedirect.com/science/article/pii/S0950584916302932 doi: https://doi.org/10.1016/j.infsof.2016.10.005

Guerron, X., Abrahão, S., Insfran, E., Fernández-Diego, M., & González-Ladrón-De-Guevara, F. (2020). A taxonomy of quality metrics for cloud services. *IEEE Access*, *8*, 131461-131498. doi: 10.1109/ACCESS.2020.3009079

Iorga, M., Feldman, L., Barton, R., Martin, M. J., Goren, N., & Mahmoudi, C. (2018). *Fog Computing Conceptual Model: Recommendations of the National Institute of Standards and Technology* (Tech. Rep.). USA. Retrieved from https://doi.org/10.6028/NIST.SP.500-325 doi: 10.6028/NIST.SP.500-325

ISO/IEC 25010. (2011). *ISO/IEC 25010:2011, systems and software engineering — systems and software quality requirements and evaluation (square) — system and software quality models.*

ISO/IEC 25040. (2011). *ISO/IEC 25040:2011, systems and software engineering — systems and software quality requirements and evaluation (square) — evaluation process.*

Katsaros, G., Kousiouris, G., Gogouvitis, S. V., Kyriazis, D., Menychtas, A., & Varvarigou, T. (2012). A self-adaptive hierarchical monitoring mechanism for clouds. *Journal of Systems and Software*, *85*(5), 1029-1041. Retrieved from https://www.sciencedirect.com/science/article/pii/S0164121211002998 doi: https://doi.org/10.1016/j.jss.2011.11.1043

Keller, A., & Ludwig, H. (2003). The WSLA framework: Specifying and monitoring service level agreements for web services. *Journal of Network and Systems Management*, *11*(1), 57–81. Retrieved from http://www.springerlink.com/index/M111304202683828.pdf doi: 10.1023/A:1022445108617

Lehmann, G., Blumendorf, M., Trollmann, F., & Albayrak, S. (2010). Meta-modeling runtime models. *Proceedings of the 2010 International Conference on Models in Software Engineering*, 209–223.

Liau, C. H., Shen, W. W., & Su, K. P. (2006). *Towards a definition of the Internet of Things (IoT)* (Vol. 60; Tech. Rep. No. 1). IEEE.

Lu, X., Yin, J., Xiong, N. N., Deng, S., He, G., & Yu, H. (2016). JTangCMS: An efficient monitoring system for cloud platforms. *Information Sciences Journal*, *370-371*, 402–423. doi: 10.1016/j.ins.2016.06.009

Modi, K. J., Chowdhury, D. P., & Garg, S. (2018). Automatic cloud service monitoring and management with prediction-based service provisioning. *International Journal of Cloud Computing*, *7*(1), 65–82. doi: 10.1504/IJCC.2018.091684

Muller, C., Oriol, M., Franch, X., Marco, J., Resinas, M., Ruiz-Cortes, A., & Rodriguez, M. (2014). Comprehensive Explanation of SLA Violations at Runtime. *IEEE Transactions on Service Computing*, *7*(2), 168–183. doi: 10.1109/TSC.2013.45

National Academies Press. (2001). *The Health of Aging Popu-*

lations. Retrieved from https://www.ncbi.nlm.nih.gov/books/NBK98373/

Runeson, P., Höst, M., Rainer, A., & Regnell, B. (2012). *Case Study Research in Software Engineering: Guidelines and Examples*.

Salesforce Company. (2020). *Heroku.* Retrieved from https://www.heroku.com/

Shatnawi, A., Orrù, M., Mobilio, M., Riganelli, O., & Mariani, L. (2018). Cloudhealth: A model-driven approach to watch the health of cloud services. *1st International Workshop on Software Health*, 40–47. doi: 10.1145/3194124.3194130

Singh, S., Chana, I., & Buyya, R. (2020). STAR: SLA-aware Autonomic Management of Cloud Resources. *IEEE Transactions on Cloud Computing*, *8*(4), 1040–1053. doi: 10.1109/TCC.2017.2648788

Stavrotheodoros, S., Kaklanis, N., Votis, K., & Tzovaras, D. (2018). A smart-home iot infrastructure for the support of independent living of older adults. *Artificial Intelligence Applications and Innovations*, 238–249.

Vora, J., Tanwar, S., Tyagi, S., Kumar, N., & Rodrigues, J. J. (2019). Hridaay: Ballistocardiogram-based heart rate monitoring using fog computing. *2019 IEEE Global Communications Conference (GLOBECOM)*, 1-6. doi: 10.1109/GLOBECOM38437.2019.9013774

Walderhaug, S., Mikalsen, M., Salvi, D., Svagard, I., Ausen, D., & Kofod-Petersen, A. (2012). Towards Quality Assurance of AAL Services. *Stud Health Technol Inform*, *177*, 296–303.

Wang, W., Feng, C., Zhang, B., & Gao, H. (2019). Environmental monitoring based on fog computing paradigm and internet of things. *IEEE Access*, *7*, 127154-127165. doi: 10.1109/ACCESS.2019.2939017

Yang, G., Jiang, M., Ouyang, W., Ji, G., Xie, H., Rahmani, A. M., … Tenhunen, H. (2018). Iot-based remote pain monitoring system: From device to cloud platform. *IEEE Journal of Biomedical and Health Informatics*, *22*(6), 1711-1719. doi: 10.1109/JBHI.2017.2776351

Yoo, B., Muralidharan, S., Lee, C., Lee, J., & Ko, H. (2019). Klog-home: A holistic approach of in-situ monitoring in elderly-care home. *2019 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, 390-396. doi: 10.1109/CSE/EUC.2019.00080

About the authors

Priscila Cedillo is an Associate Professor at the University of Cuenca-Ecuador, since 2009. She received a Ph.D. in Computer Science from the Universitat Politècnica de València (UPV) in 2017. She obtained research grants from the Senescyt for her doctoral studies and from the Fundación Carolina for a postdoctoral research stay at the UPV. She received two master's degrees, the former in Telematics from the Universidad de Cuenca, and the second in Software Engineering, Information Systems, and Formal Methods from the UPV. Her main research interests include model-driven engineering, cloud computing, software quality, and the Internet of Things (IoT). You can contact the author at priscila.cedillo@ucuenca.edu.ec.

Silvia Abrahão is an Associate Professor (accredited to Full Professor) at the Universitat Politècnica de València, Spain. She has (co)authored over 150 peer-reviewed publications. Her main research interests include quality assurance in model-driven engineering, the empirical assessment of software modeling approaches, software quality, the integration of usability/UX into software development, and cloud services monitoring and adaptation. You can contact the author at sabrahao@dsic.upv.es.

Emilio Insfran is an Associate Professor in the Department of Computer Systems and Computation at the Universitat Politècnica de València, Spain. His research interests are cloud service architectures, DevOps, model-driven development, requirements engineering, and software quality. He has published more than 150 journal and conference papers. He has contributed to more than 20 national and international research and technology transfer projects, often as principal investigator or project lead. You can contact the author at einsfran@dsic.upv.es.