CloudHealth: A Model-Driven Approach to Watch the Health of Cloud Services

Anas Shatnawi
University of Milan-Bicocca
Milan, Italy
anas.shatnawi@unimib.it

Matteo Orrù
University of Milan-Bicocca
Milan, Italy
matteo.orru@unimib.it

Marco Mobilio
University of Milan-Bicocca
Milan, Italy
marco.mobilio@unimib.it

Oliviero Riganelli
University of Milan-Bicocca
Milan, Italy
oliviero.riganelli@unimib.it

Leonardo Mariani
University of Milan-Bicocca
Milan, Italy
leonardo.mariani@unimib.it

ABSTRACT

Cloud systems are complex and large systems where services provided by different operators must coexist and eventually cooperate. In such a complex environment, controlling the health of both the whole environment and the individual services is extremely important to timely and effectively react to misbehaviours, unexpected events, and failures. Although there are solutions to monitor cloud systems at different granularity levels, how to relate the many KPIs that can be collected about the health of the system and how health information can be properly reported to operators are open questions.

This paper reports the early results we achieved in the challenge of monitoring the health of cloud systems. In particular we present CloudHealth, a model-based health monitoring approach that can be used by operators to watch specific quality attributes. The CloudHealth Monitoring Model describes how to operationalize high level monitoring goals by dividing them into subgoals, deriving metrics for the subgoals, and using probes to collect the metrics. We use the CloudHealth Monitoring Model to control the probes that must be deployed on the target system, the KPIs that are dynamically collected, and the visualization of the data in dashboards.

KEYWORDS

Monitoring, cloud service, monitoring model, quality model, software health, metrics, KPI

1 INTRODUCTION

During the last decade, the cloud computing paradigm had a pervasive diffusion, and this is witnessed by its huge impact on the information and communication technology landscape, with a revenue for the total market of public services that is estimated to be of 411.4 USD Billions worldwide by the end of 2020, with an increment of 105.6 USD Billions in the next two years [18].

SoHeal'18, May 27-June 3, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5730-2/18/05…$15.00
https://doi.org/10.1145/3194124.3194130

By allowing users (both companies and individuals) to have access to computational resources in flexible, scalable and almost ubiquitous way the cloud-based systems actually contributed to reshape the way IT companies of almost any industries run their business. As a matter of fact, many economic subjects are increasingly turning their old fashion distributed infrastructure in a new cloud-based one.

Recent technological trends seem to be addressed at overcoming the traditional IaaS paradigm [9] toward a more mature PaaS framework [16], aiming at further enhancing cloud versatility and scalability. In particular, the cloud is evolving toward a platform for the *universal connectivity* of a variety of *actors*, no matter if human or not (e.g., robots, devices, sensors, etc.), crossing the border that separates different realms (e.g.,. mobile, IoT, Telco, etc.) [5]. In this scenario where high adaptability and full integration of a large number of different services and actors in the same cloud infrastructure is demanded, requirements such as *configurability* and *programmability* are prominent. However, the high flexibility of the cloud platform must not compromise the *general health* of the systems, which have to satisfy strict reliability and availability requirements.

Addressing this trade-off between flexibility and health is particularly complex in cloud systems because it requires controlling and observing the behaviour of a plethora of sub-systems of different kinds and origin, including both virtual and physical equipments, and dealing with a large variety of subjects (e.g., business actors such as cellular, network and cloud operators, network equipment vendors, application developers) each one with different goals and objectives. While there are a number of engineered solutions to control resource and service allocations in the cloud [42], continuously checking the health of a cloud system is still an open issue. There are already solutions to collect Key Performance Indicators (KPIs) of any type from any resource and service, and there is also the technology to dynamically deploy the probes that can collect these KPIs. However, there is little work about linking these KPIs together into a *general framework* that can *visualize the health status* of both the *system* and its *individual components* based on *operator-specific* objectives.

Capturing the health status of a cloud system poses several challenges: (i) relating the concept of healthiness to actual KPIs in a way that is satisfactory for every actor involved in cloud systems, (ii) reporting the information about health and corresponding KPIs

appropriately to cloud users, and (iii) being able to dynamically reconfigure the concept of health of the system, the KPIs that must be collected, and the deployed probes, based on dynamically emerging needs.

Note that observing the health of the system is not only important for cloud operators, who can manually intervene as soon as problems are detected, but is also necessary to many approaches for self-healing and automatic program repair [4, 19, 20, 35, 39].

In this paper we present the initial results that we obtained in the design of *CloudHealth*, a model-driven approach for monitoring the health of cloud systems. CloudHealth uses a model inspired by the ISO 25011 standard [25] to encode the concept of health of a cloud system and to relate this concept to metrics that can be measured on the actual system. This model is used to control the probes that must be deployed on the target system, the KPIs that are dynamically collected, and the visualization of the data in dashboards.

Interestingly, this model also represents the basis for the definition of a common language that can improve the communication between the many parties involved in the operation of a cloud system. Although we provide an initial definition of this model, the model can be changed at any time to precisely capture the goals and best-practices of specific sub-communities and sets of operators. For instance, what the health of a system is may depend on the stakeholders and the business goals of the operators.

The paper is organized as follows. Section 2 overviews the CloudHealth approach. Section 3 presents the CloudHealth Monitoring Model. Section 4 describes the approach in details. Section 5 illustrates some use cases. Section 6 discusses related work. Section 7 provides final remarks.

2 CLOUDHEALTH

CloudHealth aims at providing cloud users with a solution to configure, deploy and operate the monitoring infrastructure. CloudHealth provides *data probes*, *data collection* and *data analysis* as a service that collectively result into a built-to-order monitoring environment. As shown in Figure 1, the CloudHealth process is structured in three main stages.

In the *Configure* stage, the cloud operator selects the *monitoring goals*, that are, the quality attributes that must be computed and observed during operation, and the relevant cloud *services*, that are, the set of services that must contribute to the quality attributes. For example, the cloud operator may decide to monitor the reliability and the efficiency of a subset of the services available in the target platform.

The choice made by the operator is automatically *mapped* into a set of metrics that must be collected from the selected services. The mapping is achieved by using the *CloudHealth Monitoring Model*, which specifies how high level quality attributes can be measured from metrics monitored on individual services. For example, if we consider *reliability* as a quality attribute, this is characterized by three sub-attributes such as continuity, recoverability and availability. The latter, for example, can be analyzed by monitoring metrics such as the *duration of failures and the number of failure occurrences*.

In the *Deploy* stage, the list of metrics that must be collected are mapped to a set of probes to be deployed on the target system based on the knowledge of the *architecture* of the system and a *probes catalog* with all the probes that can be deployed. For example, if the services to be monitored are executed on virtual machines and the time required to recover from failures must be measured, a probe that performs heartbeat pings (such as pings via ICMP or TCP) can be used. The identified probes can then be deployed and *attached* to the monitored services.

In the *Operate* stage, a dashboard is automatically *reconfigured* according to the goals selected by the operator that can be used to watch the health of both the system and its services. Operators can keep track of the system behaviour by looking at a series of different kind of visualizations, at different level of granularity (goal, sub-goals, cloud properties). For instance, the operator who is interested in monitoring system performance can visualize not only if the system performance is under control, but also the different component of Performance, all the way down to the leaves, where we find Throughput, Latency and Response Time. The dashboard is the starting point for problem diagnosis. For instance, if a monitoring goal takes an unhealthy state, the related KPIs, which represent software metrics measured on specific resources, can be shown for further analysis.

Note that after the initial selection of the quality attributes and the services to be observed, the rest of the process can be performed automatically. In this way, CloudHealth can let monitoring tools play an important role in supporting the management of cloud systems to align them with real business needs by providing a user-centered monitoring view that can flexibly and easily change over time.

3 CLOUDHEALTH MONITORING MODEL

The aim of the CloudHealth Monitoring Model (CHMM for short) is to represent the relations between high level monitoring goals and the actual software metrics that can be collected from cloud services. Cloud operators can exploit high-level monitoring goals to quickly and easily determine the aspects that must be taken under control using the dashboard. The mapping to low level metrics facilitates and automates the deployment of the probes. The model also specifies how the high-level monitoring goals can be computed from the individual metrics collected at runtime to enable the inspection of the data at different levels of abstractions.

In order to define the CHMM, we have been inspired by the ISO/IEC 25010:2011 [24] and ISO/IEC TS 25011:2017 [25] standards. We selected these standards because they provide *de facto* specifications of quality models to evaluate the quality of general IT services and they can be easily adapted to cloud services. In practice, we identified the quality attributes of the standards that can be used as monitoring goals by cloud operators. Then, we followed the standards to refine these goals into subgoals that can be mapped into measurable cloud service proprieties. We finally identified the metrics that can be used to measure these properties.

Defining Monitoring Goals. Monitoring goals are high level monitoring objectives that can be selected by cloud operators (e.g., Performance). We identify the monitoring goals by studying three quality models offered by the ISO standards: (i) the *IT service quality*, (ii) the *product quality* and (iii) the *quality in use* models. Monitoring goals are identified based on quality attributes that could be

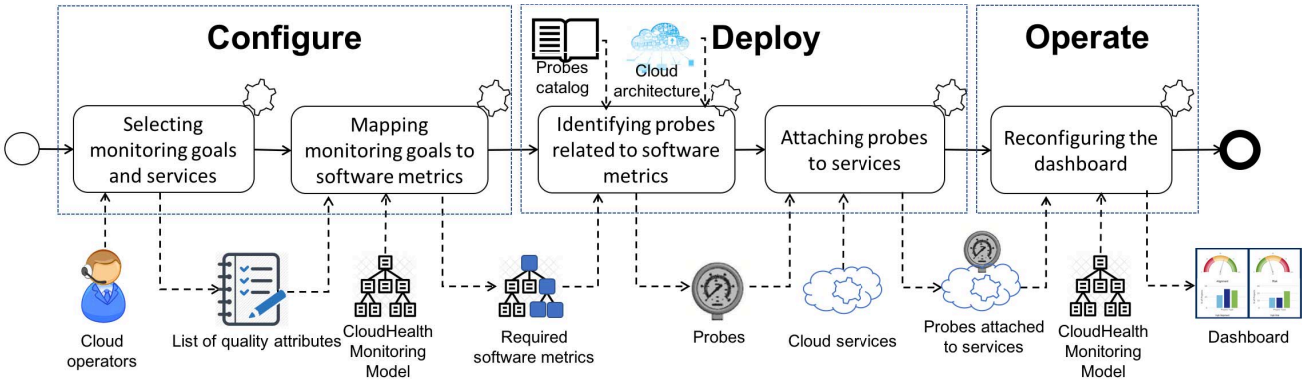

Figure 1: The CloudHealth monitoring process.

adapted for cloud services. We examined each quality characteristic based on the definition offered by the ISO standards and selected seven monitoring goals based on seven quality attributes which are *Reliability, Responsiveness, Adaptability, Effectiveness, Efficiency, Compatibility,* and *Performance.*

It is worth mentioning that other quality attributes reported in the ISO standards have not been included in CHMM. This might be due to the impossibility to adapt the attributes to cloud services (e.g., *Freedom From Risk*, *Customer Satisfaction*) or to the impossibility to measure the attribute at runtime, such as for static attributes about the internal structure of the services (e.g., *Maintainability*).

Defining Monitoring Subgoals. Monitoring subgoals are high level monitoring objectives that represent the decomposition of monitoring goals. The ISO standards provide a refinement of quality attributes into quality sub-attributes that help to support the definition of monitoring subgoals. We studied the definitions of these sub-attributes offered by the mentioned standards to define subgoals, then we identified subgoals that are measurable in the context of cloud services. Considering the *Reliability* of a service as an example, we found that *Continuity*, *Recoverability* and *Availability* are subgoals that indicate to which degree a service provides its outcomes *consistently and stably*. With regards to *Effectiveness* and *Efficiency* monitoring goals, the ISO standards do not refine them into quality sub-attributes. Thus, we did it by ourselves, adding extra subgoals that are not mentioned as quality sub-attributes in the ISO. Table 1 shows the subgoals of the seven monitoring goals coupled with their definitions.

Defining Measurable Cloud Properties. Cloud properties, such as *memory and CPU consumption*, are measurable characteristics of a cloud system, that is, it is possible to measure them using probes attached to services. We mapped each monitoring subgoal to one or more measurable cloud properties. We identified cloud properties related to each subgoal based on our experience on cloud systems. For example, the *Recoverability* of a cloud service is related to two proprieties that are (i) the time required to recover the service in case of failure, and (ii) the consistency of replicas of the service in terms of data and functions. We illustrated the resulting CHMM in Figure 2 in a hierarchical diagram which shows the elements involved in the model and their relationships.

Since CHMM is intended to cover general monitoring goals, there might be the case that some operative scenarios are not covered. The model is thus a flexible artifact that can be modified by cloud operators to adapt the CHMM to their needs. In other words, an operator can modify or add new monitoring goals and sub-goals and update the mapping of sub-goals to other cloud properties.

4 MODEL-DRIVEN MONITORING PROCESS

In this section we describe the CloudHealth process sketched in Figure 1 more in details.

4.1 Selecting Monitoring Goals and Services

Monitoring goals selection is a critical step and requires a certain level of knowledge of both the system and its services. Different services might differ in terms of quality demands which corresponds to different KPIs. For example, VoIP services are usually characterized by stringent requirements in terms of jitter, but are less demanding with respect to bandwidth. On the other hand, video broadcasting services are quite demanding also in terms of bandwidth [15]. Depending on the services that the operator decides to monitor to watch the health of the system, different sets of KPIs would be collected. To exemplify our approach, we will focus the discussion on the performance aspects (see *Performance* node in CHMM in Figure 2) in the next sections.

4.2 Mapping Monitoring Goals to Software Metrics

Once decided the set of interesting monitoring goals for a given operational scenario, the second step consists in determining the low level metrics that must be collected to measure the monitoring goals. CloudHealth provides the metrics associated to the selected monitoring goals automatically.

In principle, the process of deriving the metrics from the monitoring goals only requires visiting the tree represented in Figure 2, from the selected nodes to the leaves. As mentioned above, the model is based on ISO standard, which makes it sound. At the same

Monitoring goal	Subgoal	Definition
Reliability	Level of consistency and stability of service outcomes.	
	Continuity	Level of continuity of a service delivered under all expected circumstances, including an acceptable level of mitigation of service interruption risk.
	Recoverability	Level to which a system can restore services (including functions and data) and making them available, in the event of interruptions, failures or disasters.
	Availability	Level of service availability when it is needed.
Responsiveness	Level to which the service promptly and timely responds to requests and delivers the required functionalities.	
	Timeliness	Level to which the service timely delivers the required functionalities.
	Reactiveness	Level to which the service responds to requests in a prompt way.
Adaptability	Level of service re-adaptation to meet new needs of users.	
	Customizability	Level of service customization following its user requests.
	Initiative	Level at which a service is able to discover users' needs and consequently propose adjustments to meet these needs.
Effectiveness	Level of accuracy and completeness that a service achieves in delivering specific goals.	
	Accuracy	Level of accuracy that a service achieves in delivering deliver specific goals.
	Completeness	Level of completeness that a service achieves in delivering specific goals.
Efficiency	Level of resource consumption to deliver specific goals of users in an accurate and complete way.	
	Positive res. consumption	Resources consumed to deliver successful services.
	Negative res. consumption	Resources consumed for faulty provided services.
Compatibility	Level of compatibility of a service with other services while they share the same environment.	
	Co-existence	Level to which a service is able to efficiently deliver the required functions while it shares hardware or software resources with other services, and without impacting the other services at the same time.
Performance	Level of a service performance in relation to the amount of used resources under specific circumstances.	
	Time behavior	Level to which a service meets the requirements when it delivers its outcomes based on response time, latency and throughput rates.
	Resource utilization	Level to which a service meets the requirements when it delivers its outcomes based on the amounts and types of resources.
	Capacity	Level to which a service meets the requirements following its maximum limits.

Table 1: Monitoring Goals Model.

time, nothing hampers users from providing their own customization of the model provided that it reflects the hierarchical structure shown in Figure 2.

Getting back to our example, following the CHMM, system performance is based on three different dimensions which are: *Time Behaviour*, *Resource Utilization*, and *Capacity*. If we focus on the *Time Behaviour*, we find three metrics that should be collected: *Response time*, *Latency*, and *Throughput* [10, 27, 28, 30, 32, 33, 40, 45].

While the three metrics could already capture the time behaviour precisely, operators may decide to change the model to use a specific representation of the time behaviour. For example, the operator may decide to focus on throughput, which is a widely adopted metric that measures the quantity of input being processed by a system in the unit of time. In the network management, for example, it can be computed as the average data rate of successful data delivered over a specific communications link [15].

4.3 Identifying the Probes Related to Software Metrics

In this step CloudHealth automatically identifies the probes that must be used to monitor the identified software metrics on the selected services. For example, in case of performance, CloudHealth will identify a set of probes that can provide information about *Response Time*, *Latency* and *Throughput*. To perform this step, CloudHealth exploits information about the architecture of the cloud system and a catalog of probes that can be used to actually collect software metrics.

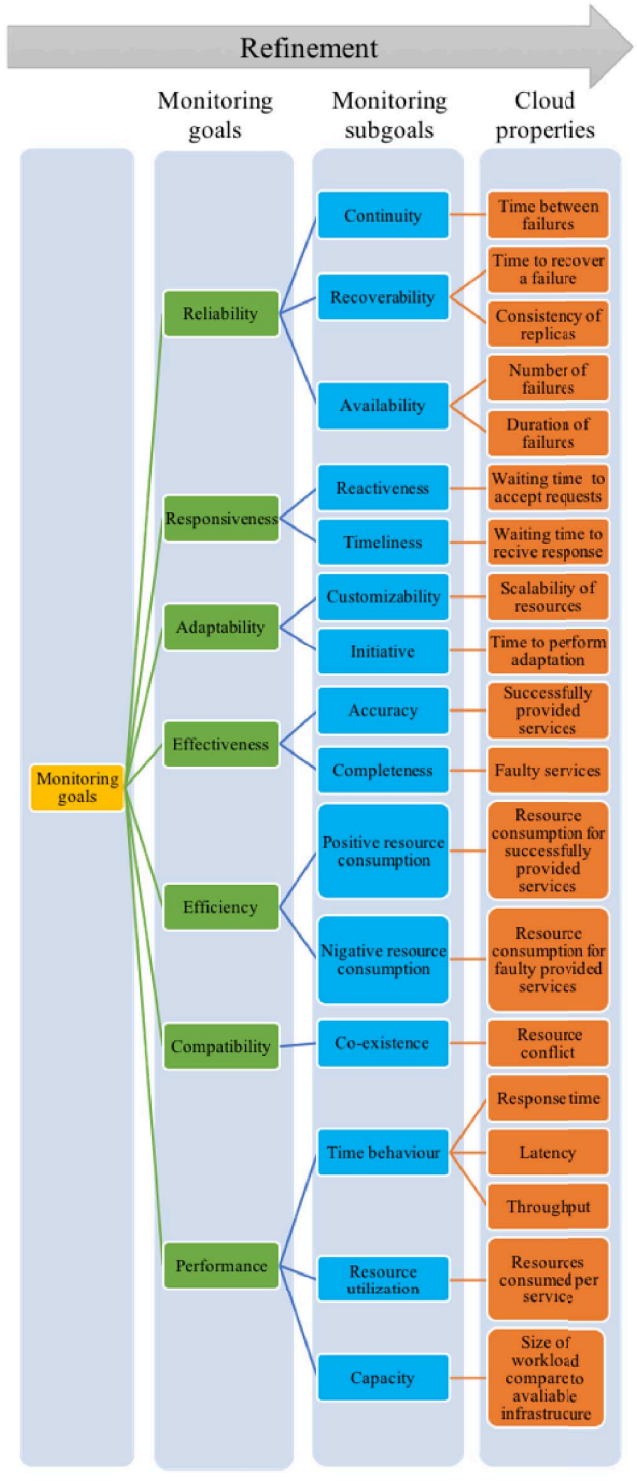

Figure 2: CloudHealth Monitoring Model.

As an example, let us consider a scenario that is becoming more and more popular, where we find a layered architecture with a PaaS (Platform as a Service) infrastructure running on top of an IaaS (Infrastructure as a Service) infrastructure, such as AWS [8] or Azure [29].

In this scenario the operators have to decide the granularity that better fits their monitoring strategies. Depending on this decision, CloudHealth selects the appropriate probes. For instance, operators may collect CPU and memory utilization at the Virtual Machine level, and thread (e.g., the number of concurrent threads active) and network activity (e.g., average request rate) at the PaaS level. CloudHealth exploits the probes catalog to select and deploy the right probes to collect metrics at the appropriate level.

It is worth noting that the performance metrics (and the related KPIs) are not independent. On the contrary, are statistically (sometimes causally) related. This implies that monitoring goals have to be considered taking care of multiple KPIs. We devise a possible CloudHealth implementation where an explicit definition of the metrics is provided, and the deployment is performed by configuration managers such as Ansible [38], Chef [12] or Puppet [36].

4.4 Attaching Probes to Services

Once the probes have been selected for deployment, CloudHealth actually deploys them. In this context we identified two situations:

(1) The monitored service and the corresponding probe are deployed at the same time.
(2) The service is already running when the probe is deployed and attached to it.

4.4.1 Deploying a service together with the corresponding probe. In this situation, the monitored service is not deployed yet. This allows to define a functional block that includes both the service and the probes. This functional block can be intended as a configuration file that specifies how the service and the probes must be configured and deployed. As an example, it could be implemented as a Cloud Package in Microsoft Azure [29], or as a Template in Google Cloud [6].

When dealing with Virtual Machines (VMs), deploying a functional block translates into creating a VM with the service and the related probe pre-installed. The deployment of the VM will result in the service running and the probe attached to it.

When dealing with containers, we may have the service as a container, thus allowing to create a new container, based on it, in which to deploy the probe. Thanks to this structure we could maintain the ability to deploy the service with or without the probe attached without having the storage overhead of keeping two full deployments as it happens with VMs.

4.4.2 Attaching a probe to a running service. If the target service is already running a simple solution could be to un-deploy it and then proceed as in Section 4.4.1. This approach however may not be optimal as it requires the service to be stopped.

A more viable solution is to firstly deploy a new instance of the monitored service together with the probe, exploit the load balancer to migrate the load to the new instance, and finally un-deploy the original service as long as it will be serving no requests.

Finally, it is possible to deploy the probe within the guest that is running the service that must be monitored. To inject the probe into the running system, runtime access to the guest is required, that is, a software agent (or an operator) should log into the guest and deploy the probe. The choice of the deployment strategy depends on the probes that must be deployed and the architecture of the cloud system.

Regardless of the deployment approach, each probe must be configured properly in order to be able to report the monitored data to the monitoring tool (e.g., an ELK installation). As an example, when deploying services and probes with an Ansible-based solution, this is accomplished exploiting *variables*, which values are then referenced in playbooks using the Jinja2 [34] templating system.

4.5 Reconfiguring the Dashboard

Operators usually do not deal directly with the choice of the software probes and consequently with the KPIs values produced by them. Instead, they indicate monitoring goals at a high level of abstraction; it is therefore feasible to assume that they might have interest in obtaining health information about the cloud system at the same level of abstraction. However the software probes provide KPIs values at the level of measurable cloud properties.

The CHMM allows to map the high level goals into cloud properties measured by collecting KPIs values. Since the CHMM model represents the relationships between the high level monitoring goals and the software metrics, CloudHealth can exploit those relationships in both directions.

CloudHealth exploits this relationship backward in order to aggregate the low level values into high level measures about the monitoring goals. As an example, consider a user that is interested in monitoring Performance. According to the CHMM, the actual software metrics collected by the monitoring framework are *Response Time*, *Latency*, *Throughput*, and *Size of the Workload*. The dashboard will however report a single score for Performance, computed according to the information in the CHMM.

CloudHealth also exploits forward relationships represented in the tree. For instance, if the operators like to know more about how the performance score is computed, they may expand the view and check the values of all the subgoals and properties that compose it. This feature may become particularly useful when an anomaly is detected in the system, as it allows one to have an overview of the situation first, but also to expand each monitoring goal to discover which cloud property collected from which service is anomalous.

In order to achieve this capability of flexibly browsing the collected information, CloudHealth reconfigures the dashboard based on the selected monitoring goals and the CHMM. In contrast to existing data visualization tools, which are inflexible, CloudHealth exploits a model-driven approach to present all and only the information relevant to the operator.

5 USE CASES

To demonstrate how to get benefits from CloudHealth, we discuss some use cases. Among several possible actors and usage scenarios, we consider two types of actors (managers and technicians) with four usage scenarios. We selected these actors due to the clear difference in their points view about monitoring goals. We will explain each use case in terms of the **Actor(s)** that will use the case, the **Wanted Feature** by these actor(s), **How** CloudHealth supports this usage scenario, and an **Example** of the use case.

Use Case 1: Monitor Services at Business Level. **Actor:** a manager. **Wanted Feature:** she wants to monitor cloud services offered by her company at high level of abstraction. This would allow her to take business decisions without delving into technical details. **How:** CloudHealth allows her to select monitoring goals at high level of abstractions (e,g., *Performance* of services) using the CloudHealth Monitoring Model. CloudHealth will automatically generate a set of probes attached to the target services to measure KPIs related to the selected monitoring goals. Next, CloudHealth will generate a customized dashboard that allows the actor to monitor KPIs related to her goals and related subgoals at different level of abstractions. **Example:** Daenerys is a manager who works at the business level and does not have knowledge about technical details. Thus, she does not understand low level KPIs, such as the *data rate of successful data delivery over a specific communications link*. She will be happy to abstract away technical details from such KPIs by providing understandable values related to monitoring goals like *Throughput* and *Performance*.

Use Case 2: Specified Monitoring for Multi-actors. **Actor:** a manager and a technician. **Wanted Feature:** they want to monitor the same service at the same time while they have different interests. **How:** CloudHealth offers a customizable model to select monitoring goals. At the technical level, it attaches the set of probes needed to collect the KPIs of interest from the selected services, regardless of the number and type of actors involved. Then, it will provide a customizable dashboard for each actor following the selected goals. **Example:** Varys is a salesman who needs to provide real evidences about the health status of the service to customers that can hardly understand low level KPIs. Varys can use CloudHealth to abstract these KPIs in a way that is understandable by these customers, for instance abstracting them to *Throughput* and *Performance* properties. Jon is a software engineer in the same company and he wants to monitor the same service at the same time. Jon is interested in low level monitoring values related to the infrastructure as his goal is to keep the service running with no single point of failure. In this context, CloudHealth provides each one a different dashboard based on their interests.

Use Case 3: Add New Monitoring Goals. **Actor:** a manager or a technician. **Wanted Feature:** they want the flexibility to redefine the hierarchical decomposition of goals into finer grained characteristics. **How:** CloudHealth is designed to be extendable, which means that the actors are allowed to add new monitoring goals. They only need to: (i) extend CHMM in terms of monitoring goals, subgoals (if exists) and cloud properties, and (ii) identify software metrics and their related probes that will be used to collect KPIs about cloud proprieties. The rest of the work will be automatically performed by CloudHealth. **Example:** Samwell is a technician who wants to monitor *Security* which is a quality attribute not included in the current version of CloudHealth. He needs to extend CHMM by defining subgoals (e.g., *Confidentiality*, *Integrity*) and maps them to cloud proprieties, software metrics and probes to collect KPIs.

Use Case 4: Dynamically Switching Between Monitoring Goals. **Role:** a manager or a technician. **Wanted Feature:** they want monitoring flexibility that allows them to dynamically switch between monitoring goals over time based on current interests. **How:** CloudHealth provides abilities to its users to switch between monitoring goals once needed. Changing the monitoring goals implies automatically deploying a new set of probes and reconfiguring the dashboard. **Example:** Petyr is interested in monitoring *Reliability* of a provided service. Once a VIP customer is being served by this service, Petyr can also extend his monitoring goals and related dashboard to include *Responsiveness* to monitoring the level that the service promptly and timely responds and provides its outcomes for that VIP customer.

6 RELATED WORK

The research described in this paper covers three distinct but related research areas: cloud quality models, cloud monitoring, and health data visualization.

Cloud Quality Models. Cloud quality models aim to assess the quality of service and build trust among cloud stakeholders [26, 44, 46]. The problem of establishing a common language between the stakeholder has been tackled by Chen at al. with a model that is characterized by three dimensions (system resource utilization, service performance, and price) described using an ontology language [13]. Wang et al. studied the relationship between user satisfaction and QoS policies, with the aim to provide a tool to adjust the utility based on the users' demands [43]. Zhou et al. proposed a hierarchical model for the evaluation of the cloud service quality. As in our case, starting from the ISO standards, the authors structured the model in six characteristics (usability, security, reliability, tangibility, responsiveness, and empathy) and sub-characteristics [47]. Adjoyan et al. defined a quality model of services by analyzing the most commonly known service definitions in the literature. Their model defined seven characteristics related to the structure and the behavior of services. They mapped these characteristics to a set of properties that can be later measured based on software metrics [3]. Other authors, focused their attention on more specific issues. This is the case of Abdeladim et al., who worked on the characterization of elasticity and scalability [1] or Ranaldo et al. who applied a semantic-based QoS model [21] to improve the brokering of resources in grid computing [37]. However these models do not deal with the actual deployment of probes for collecting the metrics, nor with the aggregation and visualisation of the monitored data.

Cloud Monitoring. Monitoring is a necessary task in cloud environments, e.g., to support capacity planning, resource management, SLA compliance and problem solving [2]. Cloud monitoring solutions range from generic tools that have been extended for monitoring cloud applications, such as Nagios [31] and Ganglia [17], to cloud-specific tools such as Amazon CloudWatch [7] and Azure Watch [29]. However, most of the current monitoring solutions focus on IT metrics rather than operator-defined characteristics and business goals. CloudHealth aims to bridge this gap with a model-driven approach. Recent results improved the support to automatic configuration and deployment of the monitoring infrastructure [11, 23]. These solutions could be integrated into CloudHealth to implement the automatic reconfiguration and deployment.

Health Data Visualization. There are many commercial and open source platforms for *health data visualization*, such as Splunk [41] and Grafana [22]. These platforms enable the construction of a number of views to present the observed data, and eventually process it. The aim is to support human understanding of the monitoring results and to facilitate decision making as a response to the observed behavior. Most of these solutions allow one to create custom dashboards even without programming knowledge [14, 22, 41]. CloudHealth is complementary to these tools since it exploits the capability to build custom views to create an organized set of views about the health of the system. The dashboard is dynamically defined using the CHHM model, which relates the health of the system to the low level KPIs that are collected.

In conclusion, there are not approaches that use models to automatically configure, deploy and operate a complete monitoring solution. The existing work concentrates on some aspects of this process only. CloudHealth represents a first step toward the definition of a complete model-driven process that starts from user needs, includes the automatic deployment of the monitoring probes, and reaches the point of generating a customized dashboard.

7 CONCLUSIONS AND FUTURE WORK

In this paper, we presented initial results in the design of CloudHealth, a model-driven monitoring solution for watching the health of cloud services. CloudHealth exploits a model, the CloudHealth Monitoring Model (CHMM), to identify the software metrics that must be collected based on the high-level monitoring goals selected by the operator. The current version of the CHMM includes 22 monitoring goals organized in a hierarchical structure, where elements at different levels of abstraction are arranged (i.e., main-goals, subgoals, etc.). To derive these monitoring goals, we started from existing standard definitions of quality characteristics that are reported in *IT service quality*, *product quality* and *quality in use* models described in ISO standards. We further refined these characteristics into measurable cloud properties and concrete software metrics to fully support automatic monitoring of the goals.

Although the current version of CHMM is not intended to be a model that fits all the cases, we design CloudHealth in a flexible way that allows cloud operators to adapt CHMM to their needs. CloudHealth uses CHMM to automatically: (i) control probes that have to be attached to cloud services based on software metrics related to the identified cloud proprieties, (ii) derive goal values from the dynamically collected KPIs, and (iii) automatically generate a dashboard that presents the data coherently with the model.

We also presented challenges – and sketch their solutions – to automatically deploy probes to running services and generate a customizable dashboard that reports KPIs for different cloud operators based on their monitoring goals. We presented four use cases of CloudHealth to show possible usage scenarios at business and technical levels.

We plan to extend our work in the following directions.

Developing a visual environment. This visual environment will allow cloud operators to work with CloudHealth at different steps, customizing their dashboards and adapting CHMM when needed.

Evaluating CloudHealth Assessing CloudHealth in real cloud environments, for instance considering both virtual machines and docker containers.

Extending CloudHealth to different scenarios Extending CloudHealth to support several domains. More precisely, we will consider the context of NGPaaS (Next Generation Platform as a Service, a running H2020 project) where three different cloud scenarios are considered: IoT, Telco and 5G.

Adapting CloudHealth to a Dev-For-Operations model. Adapting CloudHealth to support Dev-For-Operations model where multi-cloud operators are involved in both the development and operation of a cloud system.

ACKNOWLEDGMENT

This work has been partially supported by the H2020 Learn project, which has been funded under the ERC Consolidator Grant 2014 program (ERC Grant Agreement n. 646867); by the Italian Ministry of Education, University, and Research (MIUR) with the PRIN project GAUSS (grant n. 2015KWREMX), and by the H2020 NGPaaS project (grant n. 761557).

REFERENCES

[1] Alfath Abdeladim, Salah Baina, and Karim Baina. 2014. Elasticity and scalability centric quality model for the cloud. In *2014 Third IEEE International Colloquium in Information Science and Technology (CIST)*. IEEE, 135–140.

[2] Giuseppe Aceto, Alessio Botta, Walter De Donato, and Antonio Pescapè. 2013. Cloud monitoring: A survey. *Computer Networks* 57, 9 (2013), 2093–2115.

[3] Seza Adjoyan, Abdelhak-Djamel Seriai, and Anas Shatnawi. 2014. Service Identification Based on Quality Metrics Object-Oriented Legacy System Migration Towards SOA. In *SEKE: Software Engineering and Knowledge Engineering*. Knowledge Systems Institute Graduate School, 1–6.

[4] Ibrahim Al-oqily, Saad Bani-Mohammad, Bassam Subaih, and Jawdat Jamil Alshaer. 2012. A survey for self-healing architectures and algorithms. In *International Multi-Conference on Systems, Sygnals & Devices*. IEEE, 1–5.

[5] James Alleman, Paul Rappoport, and Aniruddha Banerjee. 2010. Universal service: A new definition? *Telecommunications Policy* 34, 1-2 (feb 2010), 86–91.

[6] Alphabet inc. 2018. Google Cloud Platform. https://cloud.google.com. (2018).

[7] Amazon Web Services, Inc. 2018. Amazon CloudWatch. http://aws.amazon.com/cloudwatch. (2018).

[8] Amazon Web Services, Inc. 2018. Amazon Web Services. http://aws.amazon.com. (2018).

[9] Sushil Bhardwaj, Leena Jain, and Sandeep Jain. 2010. Cloud computing: A study of infrastructure as a service (IAAS). *International Journal of engineering and information Technology* 2, 1 (2010), 60–63.

[10] Kashif Bilal, Samee U. Khan, Limin Zhang, Hongxiang Li, Khizar Hayat, Sajjad A. Madani, Nasro Min-Allah, Lizhe Wang, Dan Chen, Majid Iqbal, Cheng-Zhong Xu, and Albert Y. Zomaya. 2013. Quantitative comparisons of the state-of-the-art data center architectures. *Concurrency and Computation: Practice and Experience* 25, 12 (aug 2013), 1771–1783.

[11] Jose M Alcaraz Calero and Juan Gutierrez Aguado. 2015. MonPaaS: An Adaptive Monitoring Platformas a Service for Cloud Computing Infrastructures and Services. *IEEE Transactions on Services Computing* 8, 1 (2015), 65–78.

[12] Chef Software, inc. 2018. Chef Infrastructure Automation. https://www.chef.io. (2018).

[13] Guang Chen, Xiaoying Bai, Xiaofei Huang, Muyang Li, and Lizhu Zhou. 2011. Evaluating services on the cloud using ontology QoS model. In *2011 IEEE 6th International Symposium on Service Oriented System (SOSE)*. 312–317.

[14] Elasticsearch. 2018. Kibana: Explore, Visualize, Discover Data. https://www.elastic.co/products/kibana. (2018).

[15] John William Evans and Clarence Filsfils. 2010. *Deploying IP and MPLS QoS for Multiservice Networks: Theory and Practice*. Elsevier Science.

[16] Ana Juan Ferrer, David García Pérez, and Román Sosa González. 2016. Multi-cloud Platform-as-a-service Model, Functionalities and Approaches. *Procedia Computer Science* 97 (2016), 63–72.

[17] Ganglia. 2018. Ganglia Monitoring System. http://ganglia.info/. (2018).

[18] Gartner Community. 2017. Gartner. https://www.gartner.com/newsroom/id/3815165. (2017).

[19] Luca Gazzola, Daniela Micucci, and Leonardo Mariani. 2017. Automatic Software Repair: A Survey. *IEEE Transactions on Software Engineering* PP, 99 (2017), 1–1.

[20] Debanjan Ghosh, Raj Sharman, H Raghav Rao, and Shambhu Upadhyaya. 2007. Self-healing systems survey and synthesis. *Decision Support Systems* 42, 4 (jan 2007), 2164–2185.

[21] E. Giallonardo and E. Zimeo. 2007. More Semantics in QoS Matching. In *IEEE International Conference on Service-Oriented Computing and Applications (SOCA '07)*. 163–171. https://doi.org/10.1109/SOCA.2007.30

[22] Grafana Labs. 2018. Grafana Dashboards - discover and share dashboards for Grafana. https://grafana.com/dashboards. (2018).

[23] Hewlett-Packard Development Company. 2018. Monasca - Openstack. https://wiki.openstack.org/wiki/Monasca. (2018).

[24] ISO. 2011. *ISO/IEC TS 25010:2011(en) Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models*. ISO. International Organization for Standardization, Geneva, Switzerland.

[25] ISO. 2017. *ISO/IEC TS 25011:2017(en) Information technology — Systems and software Quality Requirements and Evaluation (SQuaRE) — Service quality models*. ISO. International Organization for Standardization, Geneva, Switzerland.

[26] Jae Yoo Lee, Jung Woo Lee, Du Wan Cheun, and Soo Dong Kim. 2009. A Quality Model for Evaluating Software-as-a-Service in Cloud Computing. In *Proceedings of the 7th ACIS International Conference on Software Engineering Research, Management and Applications (SERA'09)*.

[27] Fuliang Li, Jiannong Cao, Xingwei Wang, Yinchu Sun, and Yuvraj Sahni. 2017. Enabling Software Defined Networking with QoS Guarantee for Cloud Applications. In *IEEE International Conference on Cloud Computing, CLOUD (CLOUD'17)*.

[28] Vijay Mann, Anilkumar Vishnoi, and Sarvesh Bidkar. 2013. Living on the edge: Monitoring network flows at the edge in cloud data centers. In *2013 5th International Conference on Communication Systems and Networks (COMSNETS)*. IEEE, 1–9.

[29] Microsoft. 2018. Microsoft Azure. https://azure.microsoft.com. (2018).

[30] Jaideep Moses, Ravi Iyer, Ramesh Illikkal, Sadagopan Srinivasan, and Konstantinos Aisopos. 2011. Shared Resource Monitoring and Throughput Optimization in Cloud-Computing Datacenters. In *2011 IEEE International Parallel & Distributed Processing Symposium*. IEEE, 1024–1033.

[31] Nagios Enterprises. 2018. Nagios - The Industry Standard In IT Infrastructure Monitoring. https://www.nagios.org/. (2018).

[32] Priyanka Naik, Dilip Kumar Shaw, and Mythili Vutukuru. 2017. NFVPerf: Online performance monitoring and bottleneck detection for NFV. In *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks, NFV-SDN 2016*. IEEE, 154–160.

[33] Valerio Persico, Pietro Marchetta, Alessio Botta, and Antonio Pescapè. 2015. Measuring network throughput in the cloud: The case of Amazon EC2. *Computer Networks* 93 (dec 2015), 408–422.

[34] Pocoo Team. 2018. Jinja2 Template Engine. http://jinja.pocoo.org. (2018).

[35] Harald Psaier and Schahram Dustdar. 2011. A survey on self-healing systems: approaches and systems. *Computing* 91, 1 (jan 2011), 43–73.

[36] Puppet. 2018. Puppet - Automation for the modern enterprise. https://puppet.com. (2018).

[37] Nadia Ranaldo and Eugenio Zimeo. 2008. *A Framework for QoS-based Resource Brokering in Grid Computing*. Birkhäuser Basel, Basel, 159–170. https://doi.org/10.1007/978-3-7643-8864-5_11

[38] Red Hat. 2018. Ansible Automation. https://www.ansible.com. (2018).

[39] Chris Schneider, Adam Barker, and Simon Dobson. 2015. A survey of self-healing systems frameworks. *Software: Practice and Experience* 45, 10 (oct 2015), 1375–1398.

[40] Thamarai Selvi Somasundaram and Kannan Govindarajan. 2011. Cloud monitoring and discovery service (CMDS) for IaaS resources. In *2011 Third International Conference on Advanced Computing*. IEEE, 340–345.

[41] Splunk Inc. 2018. Create dashboards and panels - Splunk Documentation. http://splk.it/2BPqRps. (2018).

[42] Robert Szabo, Mario Kind, Fritz-Joachim Westphal, Hagen Woesner, David Jocha, and Andras Csaszar. 2015. Elastic network functions: opportunities and challenges. *IEEE Network* 29, 3 (may 2015), 15–21.

[43] Yan Wang, Jiantao Zhou, Jing Liu, Tenghe Au, and Xiaoyu Song. 2016. A QoS Evolutionary Method of Cloud Service Based on User Utility Model. In *2016 IEEE International Conference on Services Computing (SCC)*. 571–576.

[44] P. X. Wen and L. Dong. 2013. Quality Model for Evaluating SaaS Service. In *Proceedings of the 4th International Conference on Emerging Intelligent Data and Web Technologies (EIDWT'13)*.

[45] Puheng Zhang, Chuang Lin, Xiao Ma, Fengyuan Ren, and Wenzhuo Li. 2016. Monitoring-based task scheduling in large-scale SaaS cloud. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 9936 LNCS. Springer, Cham, 140–156.

[46] Xianrong Zheng, Patrick Martin, Kathryn Brohman, and Li Da Xu. 2014. CLOUDQUAL: A Quality Model for Cloud Services. *IEEE Transactions on Industrial Informatics* 10, 2 (May 2014), 1527–1536.

[47] Ping Zhou, Zhipeng Wang, Wenjing Li, and Ning Jiang. 2015. Quality Model of Cloud Service. In *2015 IEEE 17th International Conference on High Performance Computing and Communications (HPCC), 2015 IEEE 7th International Symposium on Cyberspace Safety and Security (CSS), 2015 IEEE 12th International Conference on Embedded Software and Systems (ICESS)*. IEEE, 1418–1423.