A distributed cross-layer monitoring system based on QoS metrics models

Damianos Metallidis, Kyriakos Kritikos,
Chrysostomos Zeginis, and Dimitris Plexousakis

ICS-FORTH, Greece
{metal, kritikos, zegchris, dp}@ics.forth.gr

Abstract. Monitoring of business process workflows based on metric quality models is associated to a gap between the definitions of workflow, service and infrastructure layer quality metrics. Most monitoring frameworks rely only on a layer-specific quality model, covering, e.g., the service layer, without considering the cross-layer dependencies it might have with quality models in the rest of the layers. The novelty of this paper closes the gap between the different functional layers by defining a cross-layer dependency model indicating relationships of quality aspects from three different semantic quality models. Each of these three quality models define metrics, metric aggregations and computations for each of the separate layers. These quality models are being addressed by a continuously, yet evolving distributed monitoring system.

Keywords: quality models, cross-layer dependencies, quality metrics, semantics, computation, monitoring, aggregation

1 Introduction

In order to implement cross-organisational workflows and to realise collaborations between small and medium enterprises (SMEs) the use of Web service technology and Service-oriented Architecture (SOA) has become a necessity. Whilst, SMEs are continuously moving towards service-oriented infrastructures where applications are being modeled, the need of hosting them has raised an important issue for the quality of the underlying Cloud infrastructures. Virtualization, provided by cloud infrastructures, delegates the use of any kind of resources, such as computing environments or storage systems, to the data center's internal networks. All of the above issues raised the need for monitoring of the quality of the acquired resources and of the services offered to final users as also the workfload-based procedures used by SMEs in order to use services.

We address those aspects by specifying Quality Models (QMs) that a monitoring system could adopt so as to gather monitoring data taken from three different layers: (a) Workflow, (b) Service and (c) Infrastructure layer. State-of-the-art research is based mostly on individual layers without considering the cross-layer dependencies [17] that Quality Models (QM) might have. This leads us to propose layer-specific QMs along with a cross-layer QM catering for dependencies among layers. Intention of QMs is the specification of quality terms (e.g., quality attributes) as well as of the respective relationships between these terms. In order to specify the legitimate structure of a QM, the respective conceptualisations and all possible types of quality term/concept relationships, a Quality Meta-Model (QMM) [14] should be in place. In this paper we use OWL-Q [14] as a semantic SQMM able to define semantic QMs. While different QMMs have been proposed using different representation formalisms, ontologies seem to be the best formalism as they provide a formal, syntactic and semantic description model of concepts, properties and dependencies between concepts. Moreover, they are extensible, human-understandable and machine-interpretable and enable reasoning via using Semantic Web technologies.

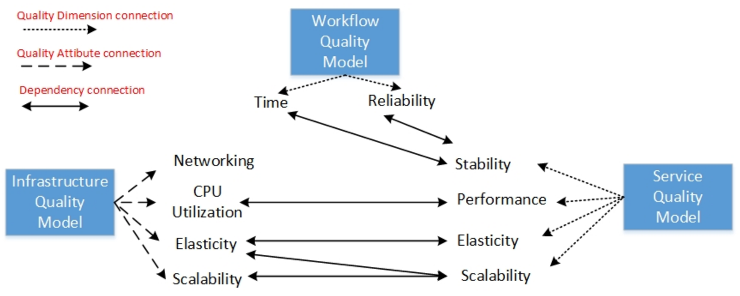

Fig. 1. Quality Models terms and cross-layer dependencies

Catering for the workflow-based usage of a service as well as the infrastructure that supports software components realizing part of the workflow functionality, we take into account three main QMs: (a) the workflow QM (WM) stressing quality terms related to tasks and workflows, (b) the service QM (SM) indicating quality terms for assessing the quality of services, and (c) the infrastructure QM (IM) encompassing quality terms suitable for the assessment of the quality of the underlying cloud infrastructure. Additionally, we have defined cross-layer relationships between quality metrics defined in the three QMs, in a sense that a metric defined in layer X can be used for the calculation of a metric defined in layer Y. Figure 1 depicts the quality models and terms that have been defined along with the cross-layer connection dependencies between them, indicating relations between quality metrics.

We are heavily focused on WM quality metrics, in a manner such that a possible user of any complicated workflow (we address this by an example of a multi-structure workflow, see section 3.1) is able to evaluate results of procedures that include benchmarking and conformance tests based on the aforementioned aspects. Following this direction we will be able to deliver Workflow *Monitor* as a Service (WMaaS) in any Workflow as a Service (WfaaS) procedure. The definitions of the quality terms that are being proposed will help SMEs to give performance and reliability grades on their business workflows, standardizing the way that the workflows are going to be evaluated.

The rest of the paper is structured as follows. In Section 2 we review the related work, while in Section 3 we represent the four QMs. In Section 4 we demonstrate the architecture of our monitoring system, whilst Section 5 provides conclusions and future work directions.

2 Related Work

As already mentioned, there are many layer-specific approaches for QMs. For instance, approaches in [12] and [7] are based on stochastic models and probabilistic theory having as a major concern the scalability of the Cloud resources based on quality metric results. Several approaches have been proposed capturing infrastructure QMs with focus on Cloud resources. Authors in [4] define QMs which support the evaluation of public Cloud services; the validation of these QMs is performed according to empirical case studies without taking into account the relations that WM, IM and SM can have between them. Precise definitions of scalability and elasticity are given in [5] and [11], spanning mostly the service and infrastructure layers; our work on the other hand defines scalability and elasticity metric dependencies between those two layers and does not cope with them in isolation.

In [6] a layer-specific extensive WM is proposed for the specification of workflow QoS, as well as methods to analyze and monitor QoS. In [10], the authors introduce the hypothesis that reliability of workflows can be notably improved by advocating scientists to preserve a minimal set of information that is essential to assist the interpretations of these workflows and hence improve their reproducibility and reusability, but with no user constraints taken into account.

A very interesting platform-independent solution has been proposed in [8] in order to support reconfiguration in service-oriented distributed soft real-time systems, in favor of time-bounded operations. As the main focus is on real-time monitoring and reconfiguration, the fact that the services might be a functional piece of a workflow has not been taken into account. This paper mainly involves service-oriented applications, which could help in the possible re-organization of the services themselves in order to function properly.

Finally, state-of-the-art research provides some approaches towards cross-layer monitoring. Kazhamiakin et al. [13] define appropriate mechanisms and techniques to address monitoring in an integrated cross-layer framework. In [9] the authors present an integrated approach for monitoring and adapting multi-layered SBAs. The approach coprises four main steps: 1) monitoring and correlation, 2) analysis of adaptation needs, 3) identification of multi-layer adaptation strategies and 4) adaptation enactment. Finally, in our previous work [17] we propose a monitoring framework for Multi-Cloud SBAs, that is able to perform cross-layer (Cloud and SOA) monitoring enabling concerted adaptation actions.

3 Definition of Quality Models

Quality dimensions cover an important aspect [14] of QMs involving dimension-specific attributes that can be measured by one or more dimension-specific metrics. Calculation formulas, quality metrics and particular types of metric relationships are defined in a formal way to formulate the structure of the respective layer. To achieve this, we have as initial guide quality dimensions/attributes, that assist in finding candidate metrics at a lower layer which could be connected/mapped to metrics at the higher layers. To produce suitable and complete QMs, which can be easily exploited by cross-layer monitoring systems, the quality terms included in them should satisfy the properties of measurability, validity and definition formalization [16]. Before we continue with the definition of the QMs to be used by a cross-layer monitoring system [17], we should also mention that these QMs rely on characteristics, stated in [15] (using OWL-Q), that each of the quality metrics should have.

3.1 Workflow Quality Model (WM)

We have defined a QM for the workflow layer, which contains workflow and task metrics as advocated in [6]. It also explicates how task measurements can be propagated to the level of workflow to produce the respective measurements of workflow metrics through metric aggregations. The proposed QM comprises the quality dimensions of *time* and *reliability*.

Time quality dimension of time is a fundamental aspect of performance that describes the time needed in order to measure, execute, record, respond and traverse through operations.

Concerning the Workflow level we have defined metrics of *Workflow Process Time* (WPT), which is calculated by the addition of the *Workflow Execution Time* (WET) and *Workflow Delay Time* (WDT) over all tasks along with the *Workflow Transition Delay Time* (WTDT) from one task to another. In addition, we have defined *Overall Workflow Execution Time (OWET)* indicating the overall execution time spent when executing all tasks in the workflow being independent from the workflow structure.

Calculation of Workflow Execution Time To assist in the calculation of WET we have defined a composite metric called *sub-WorkflowExecution* (sub-WE) which represents the execution time of a workflows sub-structure/sub-element. Possible values of the $sub - WE$ metric depend on the type of the respective sub-structure. A sub-structure could be any sequential or parallel structure within a workflow (we neglect conditional ones as they are being possessed at run-time). In the case of a sequential structure, the respective sub-WE metric is computed from the addition of Task Execution Time $TExT$, defined as the amount of time spent to perform the task by any entity (e.g., a human-based or software component), and sub-WE values mapping to the contents of this structure (i.e., tasks and internal sub-structures), respectively. In case of a parallel structure, the sub-WE metric depends on the max execution time value over the path branches included in this structure.

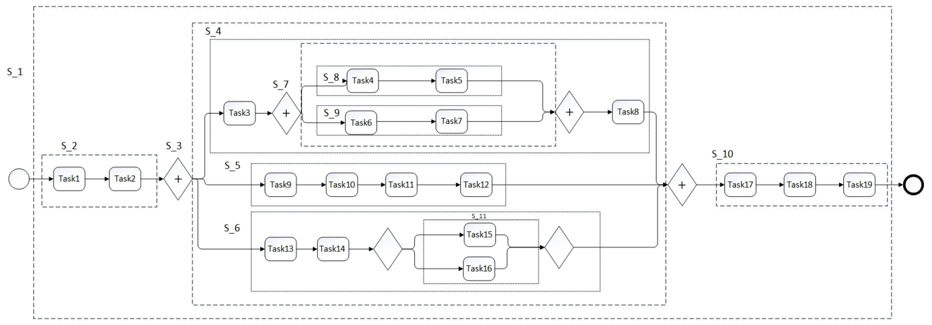

Fig. 2. BPMN Workflow indicating sub-WE elements

A new sub-WE is defined in three main situations: *(a)* For the single global/outer structure of the workflow which is equal to the workflow's WET (S_1 in Fig.2), *(b)* When next step in the workflow is a parallel structure and *(c)* when next step in the workflow is a sequential structure. A sub-WE is going to be computed once it's component values/measurements are available i.e., the encompassing sub-WE and TExT metrics.

Figure 2 depicts a nice and slightly complex BPMN workflow model which can reflect the reality and is mainly used to explicate the computation procedure for workflow metrics. This model includes eight sequential structures, two parallel structures, and a conditional structure. S symbol represents any kind of structure in the BPMN diagram. Below we represent the formulation of our algorithm along with an example following that procedure.

Metrics for the Calculation of the WET	Calculation Formulas
Workflow Execution Time(WET)	sub-WE$_1$
sub-WE$_i$	Sequential Element Case: TExT$_1$ + + TExT$_N$ + sub-WE$_2$ + + sub-WE$_K$ Parallel Element Case: max(sub-WE$_2$,...,sub-WE$_K$, TExT$_1$,...,TExT$_N$)

Table 1. Formulas for the calculation of WET.

In Table 1, i indicates a number between 1 and K, where K is the number of sub-structures which maps to sub-WE metrics and N is the number of tasks which maps to the TExT metrics.

As shown in Table 1, we rely on a procedure which leads to the production of a simple computation formula. This formula is recursively broken down into additional components, so as to compute the WET metric of a specific workflow.

To compute each of the sub-WE metrics we have the convention that sub-WE$_1$ represents the execution time spent in structure S$_1$, sub-WE$_2$ represents the execution time spent in structure S$_2$ and so on. Spanning workflow tasks, TExT$_1$ represents the execution time spent in Task1, TExT$_2$ represents the execution time spent in Task2 and so on. Thus, following the above rules and Table 1 formulations, WET equals to sub-WE$_1$, sub-WE$_1$ equals to the addition of sub-WE$_2$, sub-WE$_3$ and sub-WE$_{10}$. Sub-WE$_2$ equals to the addition of TExT$_1$ and TExT$_2$. Sub-WE$_3$ equals to the max(sub-WE$_4$,sub-WE$_5$, sub-WE$_6$). Same calculations are done for the remaining numbers of the substructures in order to get the right values. The calculation procedure of $WTDT$ and WDT metrics is similar to WET.

Reliability quality dimension corresponds to the likelihood that a component (e.g., workflow, task, service) will not malfunction or fail. We have defined the following workflow metrics:

– *Workflow Fidelity (WF)*: It measures the satisfaction of the workflow instances over the user quality requirements, within a specific time period. Thus, we can measure WF by applying the values of workflow metric measurements (mapping to the specified period of time) to a utility function which depends on users requirements.

Additionally we define the following task metrics:

– *Task Fidelity (TF)*: It computes how well tasks instances meet user requirements (at the task level) within a specific time period by utilizing similar functions with respect to the case of WF.

Workflow and Task metrics	**Calculation Formulas**
Workflow Fidelity(WF)	$f_{WF}(hist, reqs)$
$f_{WF}(hist, reqs)$	$\frac{(sat(meas_1,reqs)+...+sat(meas_A,reqs))}{A}$
$sat\,(meas_i, reqs)$	if $reqs.lowThreshold \leq meas_i.value$ $\wedge \quad meas_i.value \leq reqs.upperThreshold$ return 1; else return 0;
Task Fidelity(TF)	$f_{TF}(hist, reqs)$
$f_{TF}(hist, reqs)$	$\frac{(sat(meas_1,reqs)+...+sat(meas_B,reqs))}{B}$

Table 2. Calculation formulas of Workflow and Task metrics.

In Table 2, we define the Workflow Fidelity formula. This formula maps to the function $f_{WF}(hist, reqs)$, which takes as input two parameters. The first parameter is a set of metrics along with their measured values, mapping to the interested time interval; while the second parameter represents the user requirements. A measurement is composed of a metric, a value and a timestamp, whilst each user requirement represents a threshold being applied over a specific metric. This function has a body, which computes the mean satisfaction level of user requirements over the considered measurements represented by a value in the range [0.0, 1.0]. The satisfaction level for each measurement is computed from the *sat* function which initially selects those user requirements which map to the respective metric of the measurement and then performs the comparison of the measurement value against the user requirement low and upper bound/thresholds. If the metric measurement is within these thresholds, the output is 1; otherwise, it is equal to 0. In the case of tasks, we independently calculate their reliability in the context of a particular workflow, by considering a specific time interval and similar calculation functions to the workflow fidelity.

3.2 Service Quality Model (SM)

We have defined a SM based on metrics that assesses the quality of the respective service. Quality of a service maps to the quality that a requester perceives when using this service based on Service Level Objectives (SLOs) but also to the internal quality of this service which maps to the capturing of the service provider view. We can rely on *performance* metrics, indicating the performance of a web service; on *stability* metrics that are related to the service *reliability* and *availability*.

The *performance* dimension refers to the velocity of a service responding to any service request. It can be described by metrics that refer to the quality attribute of *Response Time*, such as *(a) Request Completion Time*, which depicts the time point that all the data mapping to a response arrive at the user, *(b) Execution Time* indicating the the time taken for the service to execute a single request and *(c) Delay Time* defining the delay time of the software component in order to start processing the request.

Stability quality dimension indicates the ability to provide reliable, continuous, consistent and recoverable services despite undesired situations like increased load, congestion, system failure and natural disasters. This quality dimension has the following quality attributes:

- *Availability*: We define availability as the ratio of time in which a service is expected to function properly. We have to stress that we do not consider networking issues as these are related to service accessibility. Service clients or third-parties assess the availability of services based on the uptime status (as conceived by these entities which could also be related to network issues that cannot be easily detected). Thus, we have defined the metrics of *Down Time* which is the total time that the service is not available and the metric of *Availability* indicating the external availability of the service as being viewed by external services. *Availability* equals to $1 - \frac{DownTime}{TotalTime}$, where $TotalTime$ is the total observation time.
- *Reliability*: The reliability quality attribute measures how reliable is the service reffering to the fact of having the least possible number of failures and the largest possible amount of time between them, according to user constraints. Metrics of this attribute are *Mean Time To Failure* (MTTF), *Mean*

Time Between Failures (MTBF) and *Fidelity* as also being defined in WM, but being adapted for the notion of SM.

By relying on [11], service *scalability* is defined as the ability of a service to scale and satisfy the agreed SLOs, when additional workload is received. The metrics defined for realizing service scalability are the *Scaling Utilization* metric which assesses the percentage of time a service exploits more or less resources than needed and *Scaling Precision (%)* which computes the percentage of time where the scaling of a service process was successful and according to the agreed SLOs, by dividing the number of successful scaling actions by the total number of scaling actions.

For the QM of Service layer we have defined the *elasticity* quality dimension as the degree at which a service system can autonomously scale the amount of service instances based on workload fluctuations to still conform to the SLOs agreed. A service's elasticity can be computed by the metrics of *Mean Time Taken to React* ($MTTrct$) and *PerfScaleFactor*. Metric of $MTTrct$ is defined as the mean time of reaction from the moment the need of scaling is detected until the respective scaling is completed, which is derived from the metric of *Reaction Time* indicating the raw time that the reaction takes. In addition, $PerfScaleFactor$ is the scale factor of the performance between two invocations of the same service before and after the scaling has been performed; equals to $\frac{\sum_{i=1}^{N} perfscalefactor_i}{N}$, where $perfscalefactor_i$ is the scale factor of i metric and N is the number of metrics being considered.

3.3 Infrastructure Quality Model (IM)

At the infrastructure layer we use four quality attributes defined in [4], as depicted in Figure 1 and define the corresponding metrics. Mainly *Network as a Service* (NaaS), *Platform as a Service* (PaaS) and *Infrastructure as a Service* (IaaS) is the reference point of most of the quality metrics defined.

Networking quality attribute characterizes the quality of a data center's (DC) network. In order to assess internally the network performance, we cover the DC operator and the SaaS provider. Both care about the DC network performance (e.g., an application execution may involve invoking different components situated in different DC VMs). We define the respective metrics of *(a) Mean Packet Loss Frequency (lost packets/min)* indicating the mean rate of lost packets that failed to arrive at their destination, *(b) Max Connection Error Rate* which defines the maximum rate at which connection errors occurs and *(c) Packet Transfer Time* indicating the mean packet transfer time from/to its source/destination accordingly.

Quality attribute of *CPU Utilization* depicts the level at which processors are leveraged within a cloud infrastructure in favour of a client. Thus, we have define the metrics of *(a) Arrival Rate (transactions/ms)* indicating the workload that is arriving at the CPU in certain point of time *(b) Shared Physical CPUs* indicating the number of different VMs function on each of the CPUs *(c) Network shared Physical CPUs* which is the number of different VMs that the physical CPU is being used of but between different network clusters *(d) Virtual CPUs* indicating the number of virtual CPUs that are being served by each of the physical CPUs and scheduled by the according hypervisor *(e) CPU Average Load (%)* which indicates the average CPU load for a processor, *(f) CPU Overall Maximum Load (%)* which indicates the overall maximum CPU Load being monitored which is further calculated by the metric of *(f_1) CPU Maximum Load(%)* indicating the peak load of the CPU within a specific period of reference. Similar metrics are being defined in case of CPU Minimum Load.

PaaS/IaaS Scalability quality attribute is defined [11] as the ability of the underlying infrastructure to sustain increasing workloads by making use of additional resources, that are directly requested, including all the hardware and virtualization layers. Based on [5], we define *Scalability Range (ScR)* as the ability of handling maximum workload that can still be handled by the underlying infrastructure to still satisfy the corresponding SLOs.

To support the notion of elasticity we are also based on [11], where it is defined as the degree to which a Cloud infrastructure can adapt on workload changes by provisioning and deprovisioning resources in an autonomic manner, such that at each time point the available resources match the user requirements, as much as possible. In addition, we rely on precision [11] to define elasticity metrics which is the absolute deviation of the according amount of allocated resources from the actual resource demand. Based on the aforementioned aspects of elasticity, we use the metrics of *(a) Precision of scaling out/scaling in* (P_O,P_I) and *(b) Mean Time To Quality Repair (MTTQR)* from [11] and [5], respectively.

3.4 Cross-Layer Dependency Quality Model

To formalize relationships between quality metrics across the three layer-specific QMs, we have considered an initial set of cross-layer dependencies in the form of a dependency quality model. These dependencies indicate (a) that the computation of a metric on layer X can be used on layer $X+1$ to complete the computation of a relevant metric and (b) the waxing dependencies that might exists between different layer metrics. Relevance could map to either both metrics belonging to the same quality dimension, or being described by similar quality attributes. Via the cross-layer dependencies (Figure 1), the metric aggregation formulas and the fact that raw quality metrics with no dependencies can be calculated by sensors placed by the distributed monitoring system on one of the respective layers, the measurability of all metrics defined is guaranteed.

We have separated the cross-layer dependencies between the metrics by considering groups of adjacent layer-specific QMs:

- For the *SM* and *WM* group the following dependencies have been captured:
    - *Task execution time* of a service task in a workflow can be computed from the execution time of the service used to realise this task's functionality.
    - A service task's fidelity equals the fidelity of the service realising its functionality, when correspondence between *task* and *service* component is valid.

  - Metric of *Task Delay Time* defined in WM can be used in order to derive the value of the *Delay Time* of a service component defined in SM.

– Next group of dependencies derived are from the SM and IM QMs:
  - *Mean Time to Quality Repair* defined in IM has an *equality* reference to SM for the $MTTrct$ defined in *elasticity* dimension. Thus, if we are referring to the same re-actions due to workload changes, then we can derive MTTrct of SM by mapping the workload changes to actions that had an impact on the scalability of the service.
  - There is an increasing trend that relates *Scaling Utilization* defined in SM and *Scaling Range* defined in IM. When utilization of scaling is high it gives us the sign that the underlying infrastructure is capable of handling scaling actions in high rates, meaning that the scaling range has also an increasing value making them waxing dependent values.
  - Regarding *CPU utilization* defined in IM we can infer that is a waxing dependent value with *response time* defined in SM. CPU utilization is increased if the overheads associated with context switching are being minimized and happen infrequently, thus large values of *CPU Average Load* for the according process context. This will have as a result the increase of the execution time for the according services.

4 Architecture of the Cross-Layer Monitoring System

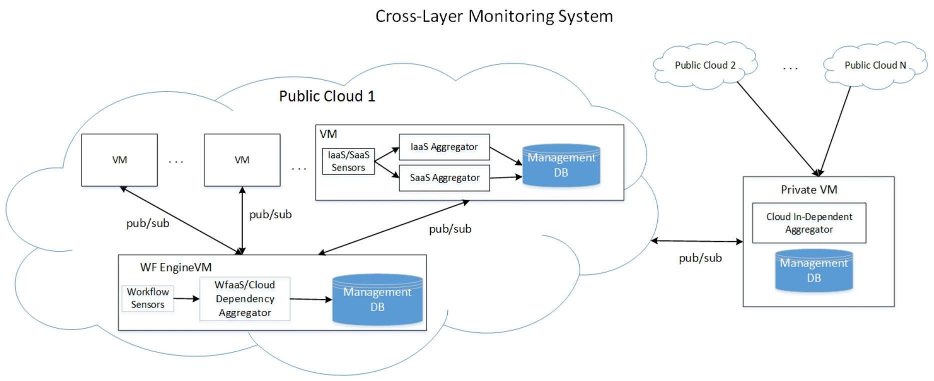

Fig. 3. Physical architecture of the Cross-Layer Monitoring System

As Figure 3 shows, Virtual Machines (VMs) in *Public Cloud 1* include monitoring sensors, metric calculation aggregators and database instances. There is more than one user VMs that compromise the IaaS and SaaS aggregators with the according sensors being deployed. Information being retrieved from the aforementioned user VMs is passed on the *WF Engine VM* through a publish/subscribe mechanism. The rationale of having only one VM that the *WF*

Engine is established on, is that a certain Workflow Engine is responsible for the business processes for each of the public clouds. Thus, by passing information to the *Workflow VM* we compute and store values of metrics related to VMs of the according Cloud. The role of the *WF Engine VM* is not only to to have a *WF Engine aggregator*, but also a Cloud-dependent aggregator that is responsible for (a) aggregation of metric values based on the cross-layer dependency QM and (b) the measurement propagation to the user VMs in order to fulfill missing cross-layer dependencies that might exist on the according user VMs.

The next step is to pass the metric dependency data, through the publish/subscribe mechanism, on our private infrastructure. Then, we store and relate metric measurements being published from different public clouds based on the QMs that we have defined. Thus, by inter-correlating metric measurements and forcing the propagation of them across public clouds we are fully aware of the functional aspect of the according services.

Monitoring tools which are being used in order to implement and extend [17] the functionality of the system are the monitoring tools of Prometheus [3] for SM, Nagios monitoring tool [2] for IM and the monitoring capabilities of Activiti [1] in case of WM. The provisioning process of the VMs in each of the public clouds sectors are out of the scope of this paper, nevertheless having a major impact on the performance of PaaS in IM.

5 Conclusions and Future Work

In this paper we have demonstrated three metric quality models along with a fourth one indicating the cross-layer dependencies and relations between quality metrics of the three QMs of WM, SM and IM. QMs that have been proposed are being covered by defining quality terms describing each of the layer's metric composability. To address that, we have created computation formulas that can be used to assign values on cross-layer depended-metrics, that could not been calculated unless dependencies are not defined. As for future work, we are in the process of extending the distributed monitoring system framework [17] in order to realize the support for the QMs proposed. Furthermore, we are going to enrich our QMs and our dependency model, so as to consider additional aspects (like security) mapping to the consideration of new domain-independent metrics.We are going to consider metrics in order to: *(a)* better validate the approach according to a specific use case; *(b)* highlight that the current structuring of the QMs is appropriate/suitable.

6 Acknowledgments

This work is supported by (a) CloudSocket project[1] that has been funded within the European Commissions H2020 Program under contract number 644690 and (b) PaaSage[2] (FP7-317715) EU project.

[1] http://www.cloudsocket.eu
[2] http://www.paasage.eu

References

1. Activiti workflow engine. http://activiti.org/.
2. Nagios monitoring tool. https://www.nagios.org/.
3. Promitheus monitoring tool. https://prometheus.io/.
4. Amid Khatibi Bardsiri and Seyyed Mohsen Hashemi. Qos metrics for cloud computing services evaluation. *International Journal of Intelligent Systems and Applications*, pages 27–33, 2014.
5. Matthias Becker, Sebastian Lehrig, and Steffen Becker. Systematically deriving quality metrics for cloud computing systems. In Lizy K. John, Connie U. Smith, Kai Sachs, and Catalina M. Llad, editors, *ICPE*, pages 169–174. ACM, 2015.
6. Jorge Cardoso, John Miller, Amit Sheth, and Jonathan Arnold. Modeling quality of service for workflows and web service processes. October 2002.
7. Jorge Cardoso, Amit Sheth, and John Miller. Workflow Quality Of Service. Technical report, LSDIS Lab, Computer Science, Universtity of Georgia, Athens GA, USA, LSDIS Lab, Computer Science, Universtity of Georgia, Athens GA, USA, March 2002.
8. Marisol Garca-Valls, Iago Rodrguez Lopez, and Laura Fernndez-Villar. iland: An enhanced middleware for real-time reconfiguration of service oriented distributed real-time systems. *IEEE Trans. Industrial Informatics*, 9(1):228–236, 2013.
9. Sam Guinea, Gabor Kecskemeti, Annapaola Marconi, and Branimir Wetzstein. Multi-layered monitoring and adaptation. In *ICSOC*, volume 7084 of *Lecture Notes in Computer Science*, pages 359–373. Springer, 2011.
10. Jos Manul Gmez-Prez, Esteban Garca-Cuesta, Jun Zhao, Aleix Garrido, and Jos Enrique Ruiz. How reliable is your workflow: Monitoring decay in scholarly publications. In Alexander Garca Castro, Christoph Lange, Phillip W. Lord, and Robert Stevens, editors, *SePublica*, volume 994 of *CEUR Workshop Proceedings*, pages 75–86. CEUR-WS.org, 2013.
11. Nikolas Roman Herbst, Samuel Kounev, and Ralf H. Reussner. Elasticity in cloud computing: What it is, and what it is not. In Jeffrey O. Kephart, Calton Pu, and Xiaoyun Zhu, editors, *ICAC*, pages 23–27. USENIX Association, 2013.
12. K. P. Joshi, A. Joshi, and Y. Yesha. Managing the quality of virtualized services. In *2011 Annual SRII Global Conference*, pages 300–307, March 2011.
13. Raman Kazhamiakin, Marco Pistore, and Asli Zengin. Cross-layer adaptation and monitoring of service-based applications. volume 6275 of *Lecture Notes in Computer Science*, pages 325–334, 2009.
14. Kyriakos Kritikos, Barbara Pernici, Pierluigi Plebani, Cinzia Cappiello, Marco Comuzzi, Salima Benbernou, Ivona Brandic, Attila Kertsz, Michael Parkin, and Manuel Carro. A survey on service quality description. *ACM Comput. Surv.*, 46(1):1, 2013.
15. Kyriakos Kritikos and Dimitris Plexousakis. Requirements for qos-based web service description and discovery. *IEEE Trans. Services Computing*, 2(4):320–337, 2009.
16. N. F. Schneidewind. Methodology for validating software metrics. *IEEE Transactions on Software Engineering*, 18(5):410–422, 1992.
17. Chrysostomos Zeginis, Kyriakos Kritikos, Panagiotis Garefalakis, Konstantina Konsolaki, Kostas Magoutis, and Dimitris Plexousakis. Towards cross-layer monitoring of multi-cloud service-based applications. In *Service-Oriented and Cloud Computing - Second European Conference, ESOCC 2013, Málaga, Spain, September 11-13, 2013. Proceedings*, pages 188–195, 2013.