2014 IEEE 13th International Symposium on Network Computing and Applications

Self-control Cloud Services

Efficient SLA Management

Tatiana Aubonnet, [1, 2] Noemie Simoni [1]
1 - Télécom ParisTech, CNRS LTCI-UMR 514, 46, rue Barrault 75634 Paris
2 - CNAM, CEDRIC, 292, rue Saint Martin, 75003, Paris
{tatiana.aubonnet, noemie.simoni}@telecom-paristech.fr

***Abstract*—We introduce a self-control integrated service component aiming at ensuring Service Level Agreement management. Our approach based on quality of service has two important points: the *contract description* (supply and demand) and the *contract management*. The *self-control* in service components allows us to react dynamically (operational decision) and the autonomic loop enables us to manage the services composition in a virtual session (tactical and strategic decision). This approach is proposed in the OpenCloudware project. We also provide an example of self-control cloud services through the Springoo application.**

Keywords-service component; self-control; SLA; service composition

I. INTRODUCTION

Today, the provisioning of a wide range of services depends on the orchestration of heterogeneous, distributed software components, which can be owned by different service providers and operate over diverse networks. In such a context, designing and providing value-added services, ensuring their nominal quality levels with service deployment, provisioning, monitoring and management becomes increasingly difficult. Provider resources must be shared by all clients. In the cloud computing context, the outsourcing introduces the need of SLA (*Service Level Agreement*). How the mapping between the provider supply and the user demand can be performed?

To answer this problem, we propose to express the SLO (*Service Level Objective*) requirements and the provided services by the same model. The main advantages of our approach are the modeling and the overall management of system behavior founded on a new integrated service component that distinguishes itself through a "self-control" property based on QoS (*Quality of Service*).

Our contribution to this problem is the use of the same modeling to which we add the following two models: *interaction behavior model* which addresses the dynamic reaction process and the *co-ordination model* further which specifies the autonomous degree of the distributed components and the overall management. According to the autonomic system concept and mutualizable component approach, we propose the management of the client session VPSN (*Virtual Private Service Network*) based on dynamic reaction in VSC (*Virtual Service Community*).

We present in this paper the feedback of our research on the self-control cloud services for efficient SLA management. This paper is organized as follows: the related works for SLA management and autonomic computing are described in Section II. Section III presents the SLA expression used in the OpenCloudware project. Section IV is devoted to our propositions for SLO, self-control service component, and SLA management. Finally, in Section V, we exhibit the advantages of our approach in a Cloud computing environment.

II. RELATED WORKS

A Service Level Agreement (SLA) is an agreement formally negotiated between two parties.

For aspects of description, we find the work of TMF (*TeleManagement Forum*) [1]. The SLA serves as a means of formally documenting the service(s), performance expectations, responsibilities and limits between cloud service providers and their users. It deals with managing service quality through the customer experience life cycle. This means managing service quality beyond the in-use phase of the life cycle in order to include sales, provisioning, in-use phase and service termination aspects. Software defined SLA offer a new design pattern that formalizes SLA and SLO as configurable parameters of cloud software components [2].

The other challenge is the minimal human oversight of the system. The work will be around the “loop of control” MAPE (Monitoring, Analyse, Planning, and Execution). For software systems, the external controller requires an explicit model of the target system in order to react to the observations and to configure and repair the system [4, 9]. Monitoring mechanisms extract and aggregate information to update the model. An evaluation mechanism detects problems in the target system as reflected in the model. The occurrence of a problem triggers an adaptation mechanism that uses the model to determine a series of actions. The mechanism then propagates the necessary changes to the target system in order to fix the problem. An external control separates the aspects of system functionality from those of adaptation behaviors [3]. But is this external MAPE loop dynamic enough? In addition we may have several causes of dysfunction not related to the same SLA?

Our motivation is to obtain efficient SLA management with the dynamic decision process and the customer virtual

978-1-4799-5393-6/14 $31.00 © 2014 IEEE
DOI 10.1109/NCA.2014.48

session management. This is why we integrate a new approach and present our proposals for the SLA contract description (supply and demand) and the SLA contract management.

III. BACKGROUND: SLA EXPRESSION

We introduce our previous propositions concerning the SLA Expression (*demand and supply*) provided in the OpenCloudware project [11]. The SLA parties represent the contracting entities of a SLA contract.

The SLA can be described in two parts:

- The users request their requirements, i.e. SLO and obligation, corresponding to the *demand*.
- The offer by Cloud provider with the guarantees provided (services offers QoS associated, penalties) corresponding to the *supply*.

On the user side, a SLO is seen as a way to express the user needs. For example: service is available 7/7 and 24/24, access time to the application < 1 s in 90% of cases, a processing time < 2s if nb req/s < 1000 in 90% of cases. The user has the obligation to check the correct functioning of the service.

On the provider side, the services offered are two types: usage and management. In accordance with our model [5], every service component integrates the QoS control. Four criteria are proposed [6] to describe the *behavior* (QoS): availability, reliability, time, and capacity.

Availability represents accessibility rate of the service component.

Reliability represents running without alteration of information (for example: error rate).

Time represents time for request processing (for example: response time*)*.

Capacity: maximum load of the service component (for example: processing capacity*)*.

These four criteria are necessary and sufficient. For each criterion the values are: design, current and threshold values.

The originality of our approach is an *SLA management* that is specific to each contract. It represents a configuration related to each user.

We analyzed this SLA expression as an input of our proposition, and in the following section we propose self-control to build cloud services compatible with the next generation services. We also present an approach for efficient SLA management.

IV. PROPOSAL: SELF-CONTROL TO BUILD CLOUD SERVICES AND SLA MANAGEMENT

In this section we present our propositions for a self-control approach. We will give our propositions for:

- Define the demand: SLO (section A).
- Define the Offer: self-control service component (section B).
- Efficient SLA management (section C).

A. *Definition of the demand : SLO (Service Level Objective)*

SLO is generally specified in terms of an achievement value or service level, a target measurement, a measurement period, and where and how to measure. For example, 90% of calls to the helpdesk should be answered in less than 20 seconds measured over a period defined as reported by the ACD system. Results should be reported by the percent of time that the target answer time was achieved compared to the desired service level (90%). The agreement between the customer and the provider on the SLO values is formalized in a SLA.

Table 1 gives an example of SLO requirements for the above services in each customer relationship steps according to the four criteria defined in our model: availability, reliability, time, and capacity. Each criterion should be expressed in SLO metrics (see four columns of the QoS criteria).

We describe the SLO related to the charging/billing service (see Table 1). The cell reference Y6-X1 (charging/billing-availability) gives the requirement regarding the availability of any type of information about cost whenever the customer requires it. SLO example: the information about cost should be accessible by different means 7 days a week and 24 hours a day.

TABLE I. MATRIX OF CUSTOMER'S SLO REQUIREMENTS

Customer relationship steps		**Detailed customer relationship steps**		**QoS criteria**			
				Availability	**Reliability**	**Time**	**Capacity**
			Cell ref.	*X1*	*X2*	*X3*	*X4*
1 – Sales			Y1				
Service management	2-Service provisioning		Y2				
	3-Service update / Technical upgrade		Y3				
	4-Service support		Y4				
	5-Repair/Trouble-shooting		Y5				
	6-Charging/Billing		Y6	SLO Y6-X1	SLO Y6-X2	SLO Y6-X3	SLO Y6-X4
	7-Cessation		Y7				
Use of Service	8- Security		Y8				
	9-Service utilization	Access	Y9.1				
		Bearer service	Y9.2				
		Service usage	Y9.3				
		User interface	Y9.4				

The Cell reference Y6-X2 (charging/billing-reliability) gives the requirement regarding the completeness and the accuracy of any type of cost information in reflecting the actual use of the service according to the conditions of the contract, in particular every tariff parameter including day time and day in the week. SLO example: the cost information available should match 100% of the costs. The Cell reference Y6-X3 (charging/billing-time) represents the requirement regarding the time measured from the end of a communication to the time when the cost information is provided to the customer. SLO example: the cost information should be supplied within less than one day after the actual cost involved. The cell reference Y6-X4 (charging/billing-capacity) represents the requirement regarding the number of customers expected to simultaneously require access to the various types of cost information. SLO example: the charging/billing service capacity should meet all simultaneous user requests.

We have described the demand expressed in SLO. In the next sub-section, we define the supply.

B. *Definition of the sypply: Self-control Service Component*

In accordance with our concepts in OpenCloudware project, we introduce a composite component based on QoS, that we call "self-control service component", see Figure 1. The main properties of our composite components are self-control, mutualization, flexibility, and exposability. *Self-control*: the composite component should have self-control and auto-survey behavior. The QoS subcomponent is enabled to conduct the monitoring of the quality of service criterion and to generate specific notifications if the threshold values are exceeded. *Mutualization*: the composite component is a multi-tenant service component. Multiple users require appeal to it at the same time. The composite component is "stateless" (it remains in the same state) in order to offer the same service to all applications. *Flexibility*: the OpenCloudware project aims to offer a toolbox of QoS components, allowing the developer (architect) to configure the desired control levels, and their distribution in the application architecture. *Exposability*: users can build their application through a catalogue or a portal.

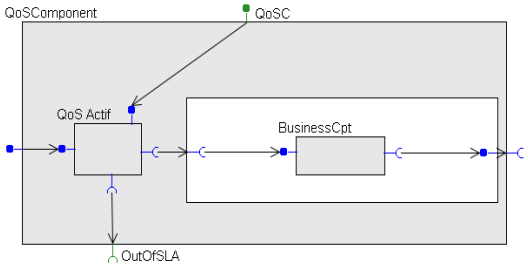

Figure 1. Self-control service component

The self-control service component contains:

- A functional content, which may be primitive or composite (BusinessCpt).
- A non-functional interface (server) QoSC, whose role is to receive configuration commands and a non-functional interface (client) OutOfSLA (send the violation indications of QoS contract).
- One or more non-functional QoS components in the membrane (the component control part), providing the capabilities of monitoring, reporting, or *QoS autonomic control*.

The component *non-functional* aspects are handled by *the component membrane*. The component QoS in the membrane plays a role of interceptor. For all the component services, incoming service requests are tested, and then the functional content of the component transmitted via the corresponding internal interfaces. The structure of our self-control service component allows us to specify precisely the non-functional information flow. We can also adapt this structure more finely, depending on the "active role" for the QoS component. The active role means that the QoS component plays the role of QoS controller, and regularly notifies the status of its QoS component. It must respect its service contract or, if it is out contract, it sends the notifications: "*out contract"*. This case is illustrated in Figure 1.

Self-control service components based on GCM (*Grid Component Model*) use an extension of the formalism ADL (*Architecture Description Language*) to take into account the presence of components in the membrane, and the rules of good composition of the bindings between the different types of interfaces. This extension is generic and allows all non-functional interface management. The GCM is a component model that was defined by the European Network of Excellence Core-Grid. It is distributed and formalized in the Grid Component Model with its hierarchical structure, encapsulation, its dynamic reconfiguration, and its non-functional controllers [7]. Figure 2 provides an example of the ADL language corresponding to Figure 1. This code is automatically generated by the Vercors tool, develop at INRIA (Sophia-Antipolis, France). The Architecture Description is based on an XML (*Extensible Markup Language*) format that contains the structural definition of the system components (subcomponents, interfaces and bindings) and some deployment concerns.

In the following section we propose efficient SLA management compatible with the objectives of self-control, i.e., dynamic reaction by the ubiquitous service components selected in Virtual Service Community; as well as the management of virtual session by VPSN (*Virtual Private Network Service).*

C. *Efficient SLA management*

The goal of this session is to present how to define an efficient SLA management between a service provider and a user. To answer this request, we present:

- Our approach (Section 1).
- Community VSC: dynamic reaction (Section 2).
- Virtual session: overall management (Section 3).

```
<definition name="QoSComponent">
    <interface signature="signature" name="S1" role="server"/>
    <interface signature="signature" name="C1" role="client"/>
    <interface signature="signature" name="QoSC" role="server"/>
    <interface signature="signature" name="OutOfSLA" role="client"/> <content>
     <component name="BusinessCpt"> <interface signature="signature" name="S1" role="server"/>
        <interface signature="signature" name="C1" role="client"/> <content class="BusinessCptClass" />
        <controller desc="primitive"/> </component> </content> <controller>
     <component name="QoSActif">
        <interface signature="signature" name="S1" role="server"/>
        <interface signature="signature" name="C1" role="client"/>
        <interface signature="signature" name="QoSC" role="server"/>
        <content class="QoS-Class" />
        <controller desc="primitive"/ </component>
     <component name="QoS"><binding client="this.S1" server="QoSActif.S1"/>
    <binding client="QoSActif.C1" server="this.S1"/>
    <binding client="this.QoSC" server="QoSActif.QoSC"/>
    <binding client="QoSActif.OutOfSLA" server="this.OutOfSLA"/> </controller>
    <binding client="BusinessCpt.C1" server="this.C1"/>
    <binding client="this.S1" server="BusinessCpt.S1"/>
</definition>
```

Figure 2. ADL description of self-control service component

1) Our approach

After having introduced our service component in the previous section, it would be interesting to explore its capabilities and contributions through the ITIL (*Information Technology Infrastructure Library*) standard [10]. We consider the following ITIL definition: (1) "To manage" is to control. (2) "To control" is to measure. (3) "To measure" is to define.

In our approach "to measure" represents the metrics of QoS criteria, "to control" represents the QoS control ("in/out contract"), "to manage" represents FCAPS (Fault, Configuration, Accounting, Performance, Security) of ISO (*International Organization for Standardization*) network management model, and MAPE-K loop. Our approach (Figure 3) allows decisions on three different levels: *strategic, tactical and operational*. The *operational* decision is based on the self-control integrated in each service component: first, we replace each service component by a dynamic reaction (i.e. by the equivalent service component in the VSC). Next, we analyze the metrics FCAPS which corresponds to "out contract". The *tactical* decision consists in the reporting of service usage and service context. The *strategic* decision is related to the strategies contained in the knowledge database.

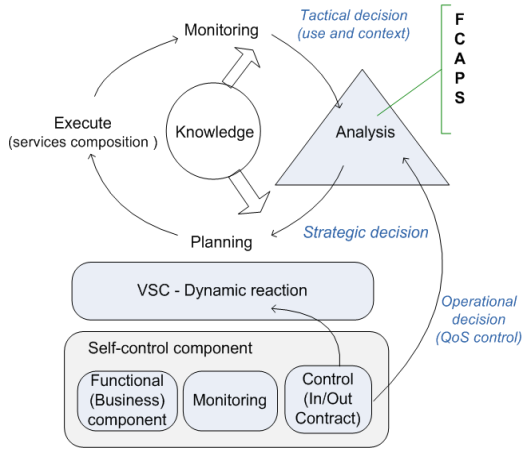

Figure 3. Decisions based on self-control component reaction

To illustrate our approach, we describe in the following section the first operational decision: the replacement of a service component that is "out contract".

2) Dynamic reaction: Virtual Service Community

The *reaction* model applies in every node and surveys the behaviors of active components; the *interaction behavior* model reflects different levels of autonomy. An active component performs a given function and may also encapsulate some resource and the operations for accessing. It is the case whenever a self-control service component can detect the problem (then it is "out contract") and finds a replacing solution, whenever an event occurs without the manager's intervention, allowing the replacement of the service component in VSC.

To allow the replacement of a service component, we use the concept of ubiquitous service. "Ubiquitous service" is a service component offering the same functionality as in the replaced component and also the same QoS described in our model, with the four criteria: availability, reliability, time and capacity. The ubiquitous service component will ensure the replacement of a degraded service component by an equivalent service in order to maintain the QoS contract. Thus, the "out contract" of service component will result in the replacement of a service element by an ubiquitous service component. This replacement is performed without breaking the session thanks to the concept of VSC.

3) Overall management: virtual session and the MAPE-K loop.

The virtual session is represented by VPSN. The Virtual Private Service Network represents the composition of the services components and their sequencing. The link represents the interactions between services at the logical level. The entities of this level provide an application service. The VPSN is created with all the service components corresponding to the commercial offer (usage, and management). In our proposition, we use an event-based approach. The self control component detects a QoS degradation and notifies

the cloud management system. The Events Manager receives the notification («out contract») and matches this event to a specific action. We introduce an MAPE-K loop for the overall management including in analysis phase a FCAPS management (Figure 3) in order to adapt the VPSN configuration.

The *co-ordination* model defines the decisions rules (tactic and strategic) more precisely, we have a decision table where for each event he indicates the service component. It is invoked in VPSN.

The contributions of our approach are definition of virtual session and the MAPE-K loop revealed suitable for the clouds self-management.

V. SPRINGOO APPLICATION

Our paper shows the advantages of self control and the SLA management through an application to Appach/Jonas/MySQL (Springoo application). It is interesting to propose an architecture that separates the two functions: *monitoring* and *control* (see Figure 4).

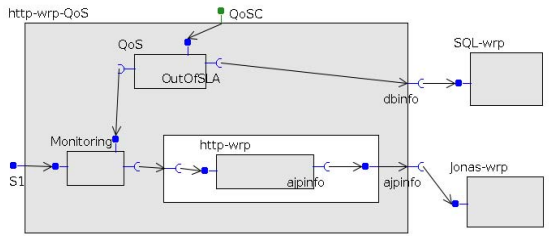

Figure 4. Springoo modelling

To illustrate the approach, the described use of Springoo case has been extended. The additional wrappers (*http-wrp-QoS, jee-wrp-QoS)*, are defined to manage the deployment of the application software components according to the QoS self-control. The *SQL-wrp* is in relation with the MySQL database server; it is an instance of the database. The *jonas-wrp*, is defined for the JEE JOnAS application server, the instance of the business logic application and the JDBC connector. Finally, the *http-wrp* manages the HTTP Apache front-end server. The *http-wrp-QoS* represents the QoS manager component of the Apache HTTP server. The *jee-wrp-QoS* represents the QoS manager components of JOnAS application server. In this modelling (Figure 4), the QoS wrapper (*http-wpr-QoS*) QoS is a composite component of type "self-control service component" containing the primitive component http-wrp. The non-functional interface *QoSC* is used to send information from the QoS as notifications ("in/out contract") or monitoring queries to database (*SQL - wrp*).

The language ADL generated in this example will be integrated in the OVF ++ (*Open Virtualization Format*) description of the Springoo application [8].

VI. CONCLUSION

In this article, we have presented an approach for a more efficient SLA management. We considered two important points to reflect the complexity introduced by the SLA/QoS management: the contract description (supply and demand) and the contract management for which we proposed a QoS model. Thus we can express the demand (SLO) across the four criteria of this model. The supplier's offer is also presented through this model.

The self-control service component allows us to react dynamically and the autonomic MAPE-K loop enables us to manage the services composition. We have proposed the management associated to self-control: first, the operational decision amongst the ubiquitous services in VSC and next, the dynamic composition of the user virtual session in VPSN. Our approach ensures that cloud users have self-control on cloud services in a dynamic way.

ACKNOWLEDGMENT

This work is supported by the OpenCloudware project funded by the French FSN (Fond national pour la Société Numérique), and is supported by Pôles Minalogic, Systematic and SCS.

We would like to thank for their help and contribution in this article: (1) OpenCloudware partners, especially Eric Madelaine and Fabienne Boyer. (2) ETSI (European Telecommunications Standards Institute) User Group especially Pierre-Yves Hébert. (3) The AIRS (Architecture et Ingénierie des Réseaux et Services) Research Group of Télécom ParisTech, particularly Ines Ayadi.

REFERENCES

[1] TeleManagement Forum GB917, "SLA Management Handbook," Release 3.0, January 2011.

[2] J. Lango "Toward Software-Defined SLAs" Published in Magazine ACM New York, NY, USA, Volume 11, pp. 54-60, November 2013.

[3] M. Amoretti, A.L. Lafuente, and S. Sebastio, "A Cooperative Approach for Distributed Task Execution in Autonomic Clouds", Parallel, Distributed and Network-Based Processing, (PDP 2013), Belfast, pp. 274 – 281, February 2013.

[4] J. Famaey, S. Latrea, J. Strassner, and F. De Turck, "A Hierarchical Approach to Autonomic Network Management," IEEE/IFIP Network Operations and Management Symposium Workshops, Osaka, pp. 225-232, April 2010.

[5] I. Ayadi, N.Simoni, and T.Aubonnet. "SLA approach for Cloud as a Service". In Proceedings of the 2013 IEEE Sixth International Conference on Cloud Computing, Santa Clara, CA, USA pages 966-967, July, 2013.

[6] T. Aubonnet, N. Simoni "Service Creation and Self-Management Mechanisms for Mobile Cloud Computing", The 11th International Conference on Wired/Wireless Internet Communications, Springer, pp.43-55, Saint Petersburg, June 2013.

[7] R. Boulifa, L. Henrio, and E. Madelaine. "Behavioural models for group communications", International Workshop on Component and Service Interoperability, number 37 in EPTCS, pages 42–56, 2010.

[8] Open Virtualization Format Specification, Distributed Management Task Force DMTF Standard, OVF 2.0, 2014.

[9] M. K. Denko, L. T. Yang, and Y. Zhang, "Autonomic Computing and Networking", book, Springer-Verlag New York Inc, November 2010.

[10] V. Lloyd "ITIL Continual Service Improvement", The Stationery Office (TSO), book ISBN: 9780113313143, August 2011.

[11] Opencloudware project, http://www.opencloudware.org/, January 2011- December 2014.