Managing the Cloud Service Lifecycle from the User's View

Beatrice Moltkau, Yvonne Thoss, Alexander Schill
Faculty of Computer Science, Technische Universität Dresden, 01062 Dresden, Germany
beatrice.moltkau@mailbox.tu-dresden.de, {yvonne.thoss, alexander.schill}@tu-dresden.de

Keywords: Cloud Computing, Service Lifecycle Management, Cloud Management System, Cloud User

Abstract: The concept of Cloud Computing has become indispensable in recent years. The use of distributed computing resources facilitates primarily infinite scalability and cost reductions by pay per use agreements. However, the management of cloud services is extensive with regard to the Cloud Service Lifecycle phases. The analysis of operational Cloud Management Systems showed that the scope of managing functionalities is too inconsistent. We present a guideline for the development of a Cloud Management System that supports the essential phases within the Cloud Service Lifecycle from the cloud provider's and the consumer's view.

1 INTRODUCTION

Cloud Computing enables the demand-oriented access to distributed computing resources (e. g. servers, storage, and services) over a broad network. The capabilities available for provisioning seem to be unlimited at any time from the user's view. The people or organizations interacting with cloud based systems can be subdivided into three core roles. The *cloud provider's* responsibilities are the provisioning, management and maintenance of the infrastructure to run the Cloud Service (CS), whereas the components are developed by the *cloud creator*. The *cloud consumer* is the one who purchases or obtains the CS. Services are in use with increasing demand in both private and public domains. Due to this quite heterogeneous target groups with respect to their interests, characteristics, goals or skills we must support and enable cloud users to take advantage of the provisioning and usage of cloud-based services. Even though the time of use of a CS corresponds to characteristic phases, the result of analyzing different existing approaches of lifecycle descriptions (Janiesch, 2011; Joshi, 2009; Brandic, 2009; Office of Government Commerce, 2007; Bizmanualz, 2011; Lee, 2007) showed that no standardized definition is available yet. Therefore, in this paper we present first of all a Cloud Service Lifecycle (CSL) that is based on the analyzed approaches in order to take advantage of their benefits. Based on this we evaluated existing operational Cloud Management Systems (CloudMS) (Amazon, 2012b; Google, 2010; Amazon, 2012a; Baun, 2011, enStratus, 2012) with respect to the required cloud user support within the lifecycle phases (Moltkau, 2012). Furthermore we present the results of this comparison. We have found out that none of these systems provide full support to cloud users. E. g., the Google Marketplace (Google, 2012) provides "insufficient" support, the AWS Management Console (Amazon, 2012a) only "sufficient" support of the lifecycle phases. None of these systems provide a satisfying assistance neither to perform contract negotiation tasks between the provider and the consumer nor to request required services adequately. Our current work focuses on providing a guideline for the development of a CloudMS that supports the entire CSL.

2 CLOUD SERVICE LIFECYCLE

To examine existing CloudMS regarding their full CSL support we have developed a CSL description (Figure 1) by combining the definitions of Janiesch (2011), Joshi (2009), Brandic (2009), the ITIL cycle (Office of Government Commerce, 2007), the PDCA cycle (Bizmanualz, 2011), and the MAPE model (Lee, 2007). The CSL consists of nine stages: Deployment, User Requirements, Matchmaking, Negotiation, Execution, Monitoring, Analyzing, Adjusting, and Ending.

A potential entry point into the cycle is the *Demand* stage. If a service with specific features does not exist its generation can be initiated here.

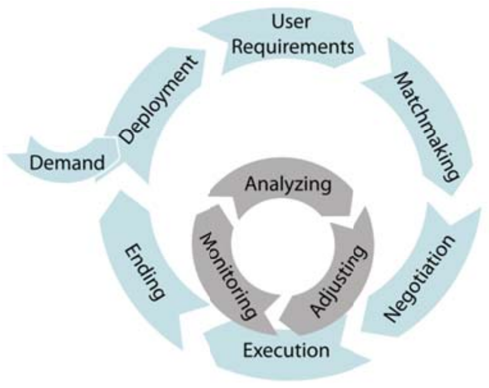

Figure 1: Cloud Service Lifecycle

In the *Deployment* stage a provider collects all information about a service in a service description. By publishing the description a service is registered and potential consumers can find it. Both stages are based on the Offering and Demand stage of the TEXO lifecycle (Janiesch, 2011) where a service can be deployed after the consumer has sent a request for it. A service is supplied to the market once all related information is published on the internet. During the next stage the consumer specifies the *User Requirements* and priorities for a service. This stage refers to the Service Requirements stage of the lifecycle model of Joshi (2009) where the consumer "details the technical and functional specifications that a service needs to fulfill". After the consumer has sent his search request a list of matching services is delivered. The *Matchmaking* process is based on the TEXO lifecycle (Janiesch, 2011) and the Service Discovery of Joshi. From the returned list of matching services the consumer picks the most fitting one. For this service he may negotiate a Service Level Agreement (SLA). First he specifies desired guarantees for the service, then the provider replies if he can fulfill them. The SLA *Negotiation* is similar to the Service Negotiation in the lifecycles (Joshi, 2009; Kümpel, 2010). When the SLA is concluded, the *Execution* stage is entered. Here the service is activated before its execution. The *Monitoring*, *Analyzing*, and *Adjusting* stages are looped through during the execution. They are based on steps from the ITIL cycle (Office of Government Commerce, 2007), the cycle of Brandic (2009), and the PDCA cycle (ISO, 2005; Bizmanualz, 2011). The CS is monitored permanently. Performance data of specific service features are analyzed. The measured values are compared with their contractually agreed value qualities. If a value is not in line with its guaranteed quality a message to the Adjusting or Ending stage is triggered to cause problem solving activities. In the Adjusting stage the infrastructure of a service can scale rapidly during runtime if necessary. In the *Ending* stage the costs for the service execution are billed, and the service is rated by the consumer (as in (Janiesch, 2011)).

3 CLOUDMS & CSL SUPPORT

From the previous CSL description we concluded requirements for existing CloudMS. We analyzed five distinct CloudMS which vary in their field of supported applications and available functionalities. They fulfill distinct requirements (Table 1).

Table 1: CloudMS support of the Service Lifecycle requirements. Key: ++ = good support, + = sufficient support, – = insufficient support, / = no support

Systems / Require-ment stage	AWS Management Console	KOALA	enStratus	AWS Marketplace	Google Apps Marketplace
Deployment	/	/	/	+	+
User Requirements	/	/	/	+	+
Matchmaking	/	/	/	+	+
Negotiation	/	/	/	–	/
Execution	+	+	+	+	/
Monitoring	++	+	++	–	/
Analyzing	++	/	++	/	/
Adjusting	++	+	++	/	/
Ending	+	+	–	+	–
Total support	+	–	–	+|–	–

The scale of support of the CS requirements reaches from "++" (full support) over "+" (fulfills more than half of the requirements) and "–" (fulfills less than half of the requirements) to "/" (fulfills none of the requirements). The first three CloudMS (Amazon, 2012a; Baun, 2011; enStratus, 2012) regulate the service infrastructure behind a service. The monitoring of the resource performance and the service execution are supported by these systems but they disregard the deployment and matchmaking of a service. The last two (Amazon, 2012b; Google 2010) are CS Marketplaces that list a choice of services available for consumers. These solutions concentrate on the preparation of a service before its execution and rarely take into account the service execution itself. The table also shows that most of the CloudMS disregard the Negotiation stage. The AWS Marketplace enables at least a choice between different payment models. The systems lack in the functionalities for payment too which leads to the insufficient support of the Ending stage.

4 GUIDELINE FOR A CLOUDMS

To remedy the problem of existing CloudMS, the missing support of the full CSL, we provide a guideline of functional requirements and a conceptual design of the system structure to enable the development of a CloudMS with full support.

The requirements catalog represents the user-oriented features a CloudMS should have based on the phases of the CSL. For service deployment the system should provide functions to specify the features and functions of the service and register it on a marketplace. To find the best fitting cloud service the consumer needs to specify threshold values and quality requirements for a service. He should also be able to set priorities for the service requirements. In the matchmaking process relevant services are selected. They should be presented to the customer in a list. One service out of the list can be chosen. The provider should be informed about a consumers request for SLA negotiation. If a cloud service provides SLA templates or ready-made license agreements they are offered to the consumer. During negotiation the service features and contract details should be presented in adapted forms for customer and provider. When both parties are satisfied with the negotiated SLA they should verify it. To start a cloud service its activation should be initialized. During the execution providers and consumers need to monitor all their registered or used services. Therefore performance data from the service is constantly requested and saved for all monitored features. The monitored data are analyzed by comparing them with the guaranteed values in the SLA. If a divergence is detected consumer and provider should be informed and a message should be forwarded for adjusting or ending the service. The adjusting component is responsible for automatic or manual changes in the service infrastructure at runtime. When an alert message is received the rules for adjusting the service infrastructure are read from the SLA or the system user can adjust the infrastructure. By adding or removing a resource the service infrastructure is scaled. As soon as an SLA expires automatically or the consumer terminates it a service is ended, its claimed resources are released and a bill for the service execution is created. The consumer also rates the service.

Contemporaneously with the requirements catalog we suggest a conceptual structure of a CloudMS with full lifecycle support. The system is structured into five functional entities. A user has to be registered with a *User Profile* to gain access to specific system areas. The *Service Registration* enables a service provider to register his services to the marketplace. Service descriptions are created at the end of the registration process. The *Service Marketplace* enables consumers to search for services matching their requirements. They can use various filtering functions and search forms. Each service has a detail page with all available information on the service. The customer negotiates his SLAs in this area. In the *Controlling and Monitoring Area* all registered services of a provider or – depending on the user profile – all used services of a consumer are listed. Services can be activated and started in the controlling views. The provider performs the SLA negotiation in this area. He can also adjust service resources. Measured data and alarm messages are displayed in the monitoring views. Services can be rated and the calculated costs for a service can be inspected here too. If changes occur the system informs the users via messages, e.g., if an alert is fired or if there are changes on the service infrastructure. All messages are saved and displayed in the *Messaging Area*.

5 EVALUATION

Based on three different use cases we have evaluated the employment of our system und have compared it with the employment of the introduced CloudMS. The use cases cover the requirements of the whole CSL. In the first use case a private consumer searches for a storage service. After contract violation the user terminates and rates the service. In the second use case a midsize company obtains office software from a provider. Since the execution of the service reaches less time than guaranteed the company decides to change the provider. In the third scenario a provider has deployed a service that is able to convert a video in various target formats. During the execution the server on which the service runs is down. Consumers whose video conversion failed get refunded. All can rate the service quality.

On the basis of our guideline we designed 14 mockups of the proposed CloudMS. We showed that one CloudMS can fulfill all requirements with regard to the lifecycle.

In the first use case our system enables the consumer to search for a storage service on a marketplace via desired requirements. After choosing a service he picks the best match and one of the ready-made contracts that are offered. The customer sees the costs for the service and can end and rate it. Compared with the other approaches our conceptual CloudMS supports the requirements of the first use case best (Moltkau, 2012). While marketplace solutions offer limited search mechanisms they rarely support the monitoring of the service execution. The provision of different agreements is not supported by any of the existing systems.

In the second use case our conceptual system can fulfill all requirements. The agreed service levels are monitored and can be inspected at. When the response time drops the customer is informed about the attempt of the provider to adjust the service infrastructure. He can end the service and rate it. During the search for a new service he can increase the priority for the response time. For the contract negotiation the customer can use a form to set his desired service quality. In comparison enStratus provides "sufficient" and therefore the best support of the existing systems. It does not support the matchmaking of a service or the rating of the service performance. None of the introduced systems support the negotiation of an SLA or the setting of priorities during the search.

The provider in the third use case can register a service by submitting a service description. When a consumer asks for executing the service the provider starts the negotiation of an SLA. During service execution the provider can adjust the infrastructure. The costs and equalization payments are calculated as well. The payment itself is handled by the system. The provider can see the ratings of his service. All requirements of the third case can be fulfilled by the conceptual system. The AWS marketplace provides "sufficient" and therefore the best support within the existing systems but cannot forward alerts, monitor the execution sufficiently, or adjust the service infrastructure. The AWS marketplace provides "sufficient" and therefore the best support within the existing systems but cannot forward alerts, monitor the execution sufficiently, or adjust the service infrastructure.

The conclusion of the evaluation is that the requirements of the three use cases could be fulfilled "excellent" by our conceptual CloudMS. The management of the last two use cases is possible in its entirety and the first use case can be managed by the system to nearly full extent. The existing CloudMS support significantly less portions of the entire lifecycle. The best result here is the support of the third use case by the AWS Marketplace. Because each of the systems lays its focus on a different area they lack functionality when it comes to full lifecycle support. The evaluation conveys that our conceptual CloudMD can not only satisfy the requirements of one specific scenario but a broad field of management requirements.

6 FUTURE WORK

The drafted CloudMS is in a very early stage. The next step is its evaluation by the target audience to examine the usability of the system. The implementation of a prototype will follow to analyze the theoretical considerations in a practical employment. We will examine if the recommendations are suitable for daily use and if all requirements are convertible. The shortcoming of the system regarding the level of detail for the search should also be improved in the future. Considerations on how the missing level of detail can be achieved have to be included into the system concept. Another improvement can be a closer connection between the system components. One example is the linking between the Service Marketplace and the Controlling and Monitoring Area. During the execution of a service the costs can be monitored to offer cheaper services with similar features.

REFERENCES

Amazon Web Services, LLC, 2012. *AWS Management Console.* (2012a), *AWS Marketplace.* (2012b).

Baun, C., Kunze M., 2011. *The KOALA cloud management service.* New York, USA

Bizmanualz, Inc., 2011. *How Are PDCA Cycles Used Inside ISO 9001?*

Bleizeffer, T., Calcaterra, J., Nair, D., Rendahl, R., Schmidt-Wesche, B., Sohn, P., 2011. *Description and Application of Core Cloud User Roles*., USA

Brandic, I., 2009. *Towards Self-Manageable Cloud Services.*

Braun, I., Reichert, S., Spillner, J., Strunk, A., Schill, A., 2008. *Zusicherung nichtfunktionaler Eigenschaften und Dienstgüte im Future Internet of Services*. PIK.

enStratus Networks, Inc., 2012. *enStratus Cloud Management Overview.*

Google, Inc., 2010. *Google Apps Marketplace.*

ISO/IEC Standard 27001, International Organization for Standardization, 2005.

Joshi, K., Finin, T., Yesha, Y., 2009. *Integrated Lifecycle of IT Services in a Cloud Environment.* NC

Janiesch, C., Niemann, M., Steinmetz, R., 2011. *The TEXO Governance Framework,* SAP Research

Kümpel, A., Braun, I., Spillner, J., Schill, A., 2010. *(Semi-)Automatic Negotiation of Service Level Agreements.*

Lee, K., Sakellariou, R., Paton, N., Fernandes, A., 2007. *Workflow adaptation as an autonomic computing problem.*, ACM

Moltkau, Beatrice, 2012. *Entwurf eines Systems zur Verwaltung von Cloud-Computing-Diensten*. TU Dresden

Office of Government Commerce, 2007. *The Introduction to the ITIL Service Lifecycle Book.*