Service Repository for Cloud Service Consumer Life Cycle Management

Hong Thai Tran[1] and George Feuerlicht [1, 2, 3]

[1] Faculty of Engineering and Information Technology, University of Technology, Sydney, hongthai.tran@uts.edu.au, george.feuerlicht@uts.edu.au
[2] Unicorn College, V Kapslovně 2767/2,130 00 Prague 3, Czech Republic,
[3] Department of Information Technology, University of Economics, Prague, W. Churchill Sq. 4, Prague 3, Czech Republic

Abstract: With rapid uptake of various types of cloud services many organizations are facing issues arising from their dependence on externally provided cloud services. In order to enable operation in this rapidly evolving environment, end user organizations need new methods and tools that support entire life-cycle of cloud services from the perspective of service consumers. Service repositories play a key role in supporting service consumer SDLC (Systems Development Life-Cycle) maintaining information that is used during the various life-cycle phases. In this paper we briefly describe service consumer SDLC and propose a design of service repository that supports information requirements throughout the service life-cycle.

Keywords: service repository, cloud services, service life-cycle

1 Introduction

Cloud computing is a novel approach for implementing enterprise IT (Information Technology) solutions that has the promise of increased agility, flexibility, elasticity and cost savings. Rapid growth in the availability of various types of cloud services provides opportunities for the implementation of innovative enterprise applications, and organizations are increasingly relying on external cloud providers to deliver a significant part of their enterprise infrastructure and applications. Unlike in on-premise situations, in cloud computing environments service consumers and service providers are typically separate entities with different roles and responsibilities during the service life-cycle. Consequently, the traditional service life-cycle used in on-premise development is not suitable in situation where cloud services are implemented by external cloud service providers and deployed by service consumers in their enterprise applications [1]. More specifically, the primary role of cloud service consumers has changed from implementation of on-premise enterprise applications to integration and management of cloud services [2], with cloud service providers taking responsibility for IT infrastructure and a significant part of the application portfolio.

The Programmable Web directory [4] currently lists almost fourteen thousand APIs (Application Programing Interfaces) for various types of services, making the identification of suitable services challenging for service consumers. In many cases, similar services are available from various cloud providers with different interfaces, protocols and Quality of Service (QoS) attributes [3]. The integration of such disparate cloud services with on-premise enterprise applications requires a significant effort. This emerging situation where enterprise applications utilize a large number of cloud services requires a new approach to service life-cycle management. A key architecture component needed to address these issues is the service repository that stores information about available services and related QoS attributes, providing a database of cloud services that are certified for use within the enterprise and can be shared among different projects.

In our earlier work [5], we have described the SDLC (Systems Development Life-Cycle) for cloud services as viewed from a service consumer perspective, and we have specified SDLC phases and described architectural components required to support life-cycle activities. This paper focuses on defining the structure and properties of the service repository. In the next section (section 2) we review research literature on service life-cycle management and service repositories. The following section (section 3) is a description of the proposed service repository structure for cloud service consumer life-cycle management, and section 4 contains our conclusions and proposals for future work.

2 Related work

The life-cycle of a cloud services involves different stakeholders that include service providers and service consumers that participate in delivering cloud-based enterprise applications and ensuring runtime management of cloud services. Generally, service life-cycle management includes three types of activities: design time, runtime and change time activities. Although cloud service life-cycle is still a subject of extensive investigation, there is a general agreement in the literature about the individual life-cycle phases and the need for a service repository to support life-cycle activities.

In early research, Yelmo, et al. [6] describe user-centric service life-cycle management for telecom services. The authors focus on Service Lifecycle Manager and the Service Execution Environment modules of the OPUCE platform (Open Platform for User-centric service Creation and Execution). In OPUCE, a service repository is used to store service description including all related attributes e.g. service type, descriptions, and the terms and conditions of use. Services are specified using three sets of *facets* (i.e. description of a specific aspect of a service): Functional facets, Non-functional facets and Management facets. Vitharana and Jain [7] introduce a Knowledge Based Component Repository (KBCR) for enabling requirements analysis. The repository includes basic information about services (name, version, functionalities, and QoS attributes), facet information, business process templates, relationships among components, and provides support for a search capability. Yu, et al. [8] propose a semantically enhanced service repository for user-centric service dis-

covery and management. The repository consists of two main components: a service registry for storing and managing service metadata (i.e. service name, service version, provider and service descriptions) and a service discovery component that allows discovery of services. Lakshmi and Mohanty [9] describe the design of a scalable service repository implemented using a relational database supporting algebraic operators for service composition using Composition Search Trees. The database service includes five tables: Providers, Services, Parameters, Service Input and Service Output. Service providers are categorized by reputation (using categories Best, Good, Average and Below Average), and services are classified using QoS attributes. This information is used to search for services in the registry and to compose business process based on identified services.

Shetty and D'Mello [3] review service repository strategies and service discovery techniques with the aim to support diversity of cloud services. The cloud service discovery feature supports search and browsing of services based on functional and non-functional properties. Authors classify discovery methods according to different architectures of the cloud service repository into centralized architectures and distributed architectures. They also describe the various service discovery algorithms used in the literature for cloud service discovery such as functional description based methods: keyword (syntactic) based discovery, semantic based discovery and hybrid matching. Non-functional description method that includes *static and dynamic QoS* based methods. A method for managing integrated life-cycle of cloud services was proposed by Joshi, et al. [10]. The authors have identified performance metrics associated with each life-cycle phase that include data quality, cost, and security metrics based on SLA (Service Level Agreement) and consumer satisfaction, and they have proposed a service repository with a discovery capability for managing cloud services life-cycle [1]. The authors divide cloud services life-cycle into five phases: requirements specification, discovery, negotiation, composition, and consumption. During the service discovery phase, service consumers search for services using service description and provider policies in a simple services database. Service information is stored as a Request for Service (RFS) that contains functional specifications, technical specifications, human agent policy, security policy, and data quality policy.

Field, et al. [11] present a European Middleware Initiative (EMI) Registry that uses a decentralised architecture to support service discovery for both hierarchical and peering topologies. The objective of the EMI Registry is to provide robust and scalable service discovery that contains two components: Domain Service Registry (DSR) and Global Service Registry (GSR). Service discovery is based on service information stored in service records that contain mandatory attributes such as service name, type of service, service endpoint, service interface, and service expiry date. Vukojevic-Haupt, et al. [12] proposed a service selection method for on-demand provisioned services. Services are provided by a third party provider and service consumers have no knowledge about the implementation and the underlying infrastructure that supports the delivery of services. Authors develop an entity relationship diagram of the service registry that contains service information and metadata, including functional and non-functional properties, service configuration parameters, service provider information, functional description of the service, and QoS attributes. In a recent pub-

lication Bauer, et al. [13] present the design of an advanced SOA repository enriched with analysis capabilities. The repository contains various types of services and their relationships. Authors propose a meta-model for repositories to analyse service dependency and the impact of changes.

Most of the research publications reviewed in this section focus on service selection and discovery. Our service repository design aims to cover the entire life-cycle of cloud services from the perspective of service consumers, and includes the phases: requirements specification, service identification, service integration, service monitoring and service optimization.

3 Repository support for service consumer SDLC

As noted in our previous work [5], traditional SOA systems development methodologies do not explicitly differentiate between service provider and service consumer SDLC cycles. In the context of cloud computing, service providers and service consumers are separate entities that perform different tasks throughout their SDLC cycles. Service providers are responsible for the implementation and delivery of cloud services and service consumers are primarily involved in the selection and integration

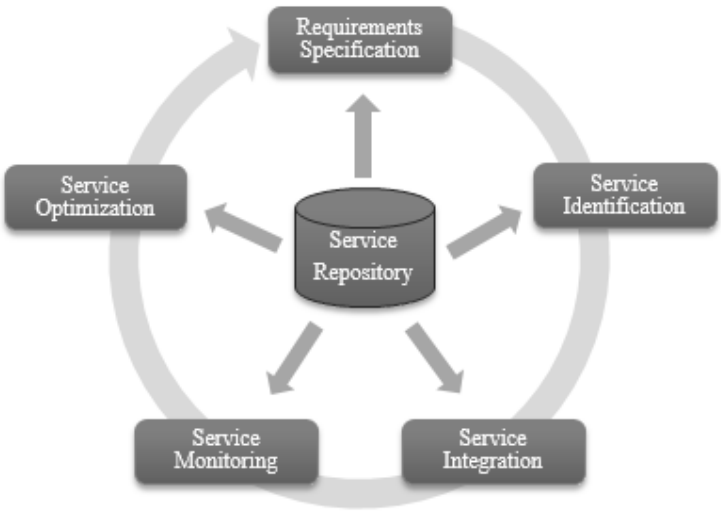

Fig. 1. Cloud service consumer life-cycle

of suitable cloud services into their enterprise applications. As illustrated in Figure 1, we identify five SDLC phases of the service consumer life-cycle: requirements specification, service identification, service integration, service monitoring and service optimization. These phases can be classified into design-time activities that include requirements specification, service identification and service integration, and run-time

activities that involve service monitoring, and service optimization. The information held in the service repository is used to manage services and to define service compositions that are executed by the workflow engine at runtime. In the following sections we consider information requirements for the individual life-cycle phases and define the structure and properties of the service repository.

3.1 Requirements Specification

The service requirements specification phase involves description of functional and non-functional requirements that a given service needs to fulfil. Functional specifications of the service describe what functions the service should provide. While there are differences in the specification according to the type of service (e.g. application service, infrastructure service, etc.), typically the specification includes technical details of the service interface (e.g. WSDL interface) and may also include details of the technological environment (e.g. specific hardware platforms, programming languages, etc. in the case of infrastructure and platform services). The non-functional attributes include service availability, response time, and security requirements, and may also include requirements regarding data location, security certification and the maximum cost of the service. Once the service is fully described and classified, the service consumer creates a Request for Service (RFS) and records the information in the service repository [10].

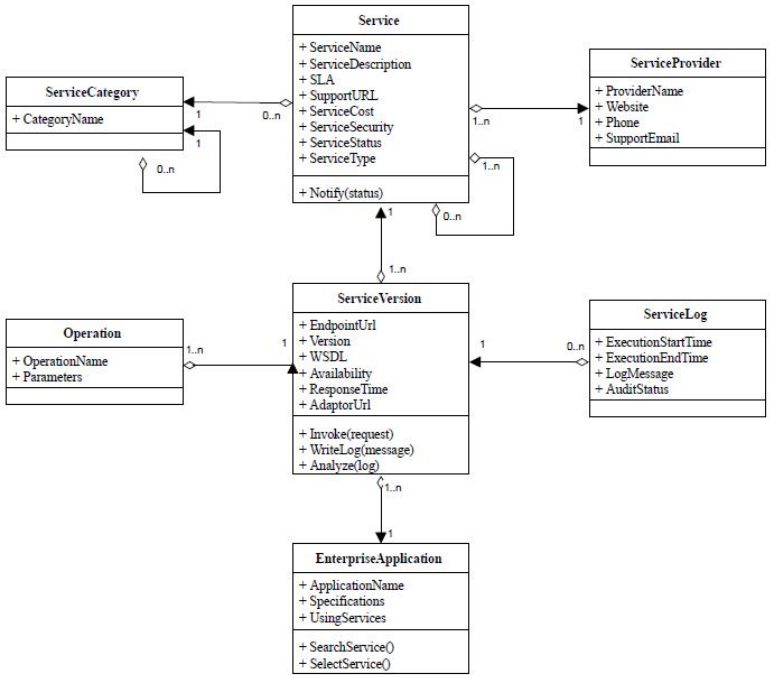

Fig. 2. UML diagram of the Service Repository

Table 1. List of repository attributes

Attribute	Description
Service	
ServiceName	The unique identifier of the service
ServiceDescription	Description of the service
SLA	Service level agreement
SupportUrl	URL of the support page of the service
ServiceCost	Cost usage plan of service
ServiceSecurity	Security characteristics of the service
ServiceStatus	Service status, e.g. online, offline or retired
ServiceType	The type of service (on-premise, cloud or composite)
Service version	
EnpointUrl	Network location of the service
Version	Service version number
WSDL	WSDL specification of the service
Availability	Service availability (estimated)
ResponseTime	Service response time (estimated)
AdaptorUrl	Network location of the service adaptor
Operation	
OperationName	Service method name
ServiceParamater	Service method parameters
EnterpriseApplication	
ApplicationName	Name of application
Specifications	Application specification requirements
UsingServices	List of services are using in this application
ServiceLog	
ExecutionStartTime	The start time of service execution
ExecutionEndTime	The end time of service execution
LogMessage	Log message (e.g. error message)
AuditStatus	Service outcome (i.e. success or failure)
ServiceCategory	
CategoryName	Service Category Name
ServiceProvider	
ProviderName	Service provider name
Website	Service home page or customer support page
Phone	Customer service hotline
SupportEmail	Customer support email

Figure 2 show the initial version of service repository UML (Unified Modelling Language) diagram, and Table 1 is a list of repository attributes derived from the UML diagram. *Service* is a central entity of service repository and includes attributes that describe registered services including service identification, a range of functional, non-functional attributes, and SLA description. In order to manage service evolution and keep track of changes of service functionality, information about *Service Versions*

is stored in the repository. *Operation* is associated with service versions as it is possible for different versions of the service to have different operations when the service evolves. *Service Category* is used to categorize services according to service type resulting in a service type hierarchy illustrated in Figure 3. The concept of service substitution is represented by the *replaces* relationship that identifies services with same functionality (e.g. two payment services with identical functionality) that provide alternatives that can be used to improve service availability, or to replace services to reduce the cost and improve performance. Service substitution information is used at design time to support load balancing and failover features. *Service Provider* represents service providers and contains service provider attributes listed in Table 1. *Service Log* records runtime information that includes response time, results of service invocation, and other non-functional attributes collected at run-time and used for analysis of service performance. Each service can be used in a number of *Enterprise Applications*, and each enterprise application can use a number of registered services.

3.2 Service identification

Service identification is constrained by the functional and non-functional requirements documented in the previous phase (requirements specification phase). Service identification phase uses service category hierarchy (Figure 3), and functional and non-functional attributes of the service identified during the service requirements phase. Service repository has a web-based user interface which allows consumers to

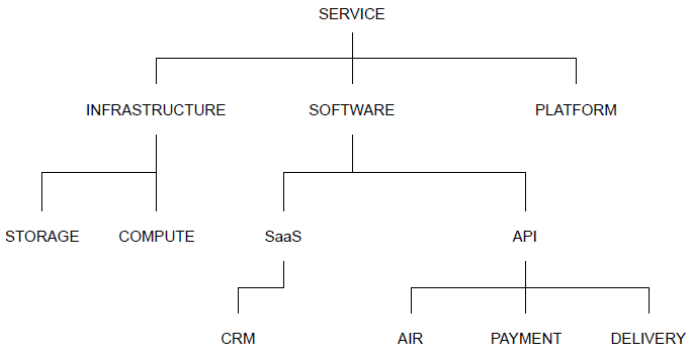

Fig. 3. Partial service category hierarchy

search for services based on their category and QoS information. Service identification phase begins by searching the service repository, attempting to match the requirements specified in the previous phase with services that are already registered in the repository and certified for use. If no existing service matches the requirements, the service consumer will need to search for the candidate services available from cloud service providers, or contact a preferred service provider directly to locate a suitable cloud service. In addition to selecting a suitable the service, the identification phase involves service testing and approval. Service approval is an internal certifica-

tion process that certifies cloud services for use in enterprise applications within the organization. Given the large number of available cloud services, the selection of suitable services can be time consuming, in particular if this task is performed multiple times in the context of different projects that require similar services. Using the consumer service repository to store information about approved cloud services ensures that services are shared among different projects, and that service selection and approval process is not unnecessarily repeated. In some instances, the consumer may be able to negotiate details of the SLA with the service provider, although this will depend on the type and volume of services involved.

3.3 Service Integration

Following the service identification phase, cloud services need to be integrated into consumer enterprise applications. Following the registration of the enterprise application, relevant services are identified and composed to implement the desired business functionality using services that have been already certified and are recorded in the repository. The service substitution information is used to compose services. The design of a composite service involves searching for atomic services that match the

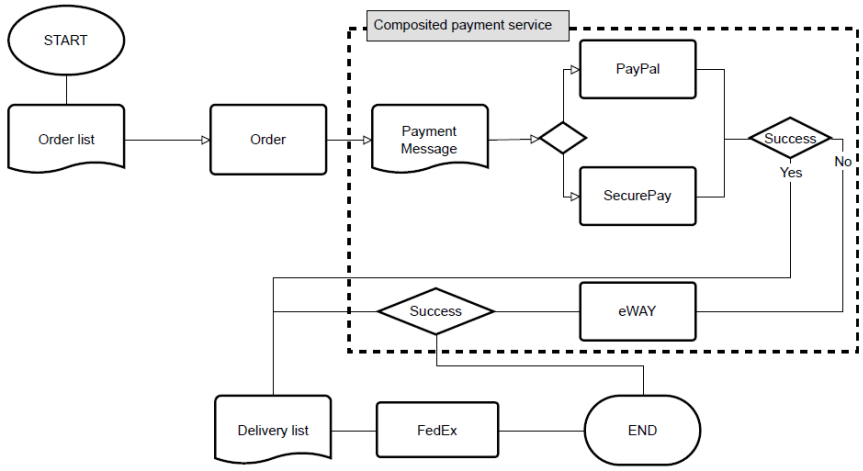

Fig. 4. Composite payment service for online shopping process

requirements of enterprise applications and composing these services to define a suitable runtime execution sequence. For example, the online shopping process illustrated in Figure 4 includes a composite payment service composed of three different (atomic) payment services: PayPal, SecurePay and eWay. This composite payment service is used to load-balance the payment services, and at the same time provides a failover function that handles situations when a particular service becomes unavailable. This improves both the availability and the reliability of the enterprise application.

3.4 Service Monitoring

The service monitoring phase involves monitoring activities that take place at runtime and includes the management of service utilization. Typically, both the service provider and service consumer perform service monitoring independently, and both parties are responsible for resolving service quality issues that may arise. The service repository includes information that records runtime performance of services (i.e. response time, availability information, and various type of error messages) generated by the Notification Centre that records service status of cloud services in the runtime service log. This information is used by application administrators to monitor service utilization, plan maintenance activities, and to perform statistical analysis of response time and throughput for individual cloud services. Maintaining accurate QoS statistics in the service repository enables to compare the values of QoS attributes defined in the SLA against the actual (measured) QoS values.

3.5 Service Optimization

Service optimization phase is concerned with continuous service improvement. This can be done by replacing existing services with new versions when these become available, or by identifying substitute services from a different provider that have the same functionality. For example, the payment service PayPal could be replaced by the SecurePay service, based on information stored in the repository during the monitoring phase. Service repository supports the process of service optimization allowing service replacement without impacting on existing enterprise applications. In addition to optimizing individual services, entire business processes can be optimized by redesigning the constituent composite services.

4 Conclusion

The main difference between service provider SDLC (i.e. traditional service lifecycle as described in the literature) and service consumer SDLC is the focus on service integration and runtime management of services. Cloud service integration is a design-time activity that relies on accurate description of service interfaces and associated QoS attributes to allow service composition and definition of service execution sequences to implement specific business functions. Run-time activities include failover management and ensuring satisfactory levels of service quality to maintain continuity of operation. To achieve these objectives, designers must be able to match desired QoS attributes values against information stored in the repository and to define processing rules that determine the sequence of service execution at run-time [14].

Well-designed service repository is critical for the support the various activities throughout the consumer service life-cycle. In this paper, we have described the design of service repository that supports the information requirements of the life-cycle phases: requirement specifications, service identification, service integration, service monitoring and service optimization. Service repository structure includes both functional and non-functional attributes allowing a full description of the service for the

purpose of creating RFS (Request for Service). Structuring service specification using service category hierarchy allows accurate matching of services based on service type and QoS attributes. During the service integration phase, service designers use this information to implement composite services with desired run-time properties (i.e. failover capability and load balancing).

In conclusion, our service repository design supports both design time and runtime activities throughout the service consumer SDLC. We are currently in the process of implementing the service repository using Microsoft SQL Server database and further enhancing the design of the repository.

References

1. Joshi, K. P., Yesha, Y., Finin, T.: Automating Cloud Services Life Cycle through Semantic Technologies, IEEE Transactions on Services Computing, vol. 7, pp. 109-122 (2014).
2. Farrell, K.: Cloud Lifecycle Management: Managing Cloud Services from Request to Retirement. http://www.bmc.com/blogs/hybrid-cloud- delivery-managing-cloud-services-from- request-to-retirement
3. Shetty, J., D'Mello, D. A.: Repository Design Strategies and Discovery Techniques for Cloud Computing, In: 2013 International Conference on Green Computing, Communication and Conservation of Energy (ICGCE), pp. 761-766 (2013)
4. ProgrammableWeb: The World's Largest API Repository, Growing Daily. http://www.programmableweb.com/apis/directory
5. Feuerlicht, G., Tran, H. T.: Adapting Service Development Life-cycle for Cloud, In: The 17th International Conference on Enterprise Information Systems (ICEIS), Spain (2015)
6. Yelmo, J., Trapero, R., del Álamo, J., Sienel, J., Drewniok, M., Ordás, I., *et al.*: User-Driven Service Lifecycle Management – Adopting Internet Paradigms in Telecom Services, In: The 5th International Conference on Service-Oriented Computing (ICSOC), Austria (2007).
7. Vitharana, P., Jain, H.: A Knowledge Based Component/Service Repository to Enhance Analysts' Domain Knowledge for Requirements Analysis, Information & Management, vol. 49, pp. 24-35 (2012).
8. Yu, J., Sheng, Q. Z., Han, J., Wu, Y., Liu, C.: A Semantically Enhanced Service Repository for User-centric Service Discovery and Management, Data & Knowledge Engineering, vol. 72, pp. 202-218 (2012).
9. Lakshmi, H., Mohanty, H.: RDBMS for Service Repository and Composition, In: The 4th International Conference on Advanced Computing (ICoAC), pp. 13-15 (2012)
10. Joshi, K., Finin, T., Yesha, Y.: Integrated Lifecycle of IT Services in A Cloud Environment, In: The 3rd International Conference on the Virtual Computing Initiative (ICVCI), USA (2009)
11. Field, L., Memon, S., Márton, I., Szigeti, G.: The EMI Registry: Discovering Services in a Federated World, Journal of Grid Computing, vol. 12, pp. 29-40 (2014).
12. Vukojevic-Haupt, K., Haupt, F., Karastoyanova, D., Leymann, F.: Service Selection for On-demand Provisioned Services, In: The 18th International Enterprise Distributed Object Computing Conference (EDOC), Germany, pp. 120-127 (2014)
13. Bauer, T., Buchwald, S., Tiedeken, J., Reichert, M.: A SOA Repository with Advanced Analysis Capabilities-Improving the Maintenance and Flexibility of Service-Oriented Applications, (2015).

14. Feuerlicht, G., Tran, H. T.: Service Consumer Framework: Managing Service Evolution from a Consumer Perspective, In: The 16th International Conference on Enterprise Information Systems (ICEIS), Portugal (2014)