ADAPTING SERVICE DEVELOPMENT LIFE-CYCLE FOR CLOUD

George Feuerlicht[1,2,3] and Hong Thai Tran[1]
[1] *Faculty of Engineering and Information Technology, University of Technology, Sydney,*
[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*
george.feuerlicht@gmail.com, tranhongthai@gmail.com

Keywords: Cloud Computing, Service-Oriented Architecture, Service Life-Cycle

Abstract: As the adoption of cloud computing gathers momentum, many organizations are facing new challenges that relate to the management of cloud computing environments that may involve hundreds of autonomous cloud services provided by a large number of independent service providers. In this paper we argue that the large-scale use of externally provided cloud services in enterprise applications necessitates re-assessment of the SOA paradigm. The main contribution of this paper is the identification of the differences between service provider and service consumer SDLC cycles and the description of the service consumer SDLC phases.

1. INTRODUCTION

Over the last decade SOA (Service Oriented Architecture) has become the architecture of choice for most large organizations, providing a flexible and responsive enterprise computing architecture that addresses the business needs of modern organizations. Large-scale adoption of SOA has been facilitated by wide industry support and high level of web services standardization resulting in comprehensive tools and methods that support the entire SDLC (Systems Development Life Cycle) of service-oriented enterprise applications. A key benefit of SOA is that it enables close alignment of IT (Information Technology) with the business objectives of the organization, and at the same time facilitates high-levels of business process automation by using discrete, loosely coupled services. Another important benefit of SOA is that it enables organizations to participate in the emerging world of cloud computing and consume software and hardware services provided by other parties. However, the recent shift towards the use of externally provided cloud services in enterprise applications has altered the basic premise of SOA and requires a re-assessment of the SOA paradigm and in particular the SOA SDLC. The underlying assumption of traditional IT architectures, including SOA, is that application services are developed and deployed on-premise by the end-user organization. More specifically, traditional SOA assumes that the life cycle of enterprise services is controlled by the end-user organization, i.e. that most of the services are designed, developed and provisioned by the same organization that deploys the services in its enterprise applications. Although, SOA services can be consumed over the Internet or a private network from remote locations (e.g. using SOAP or REST protocols), the primary focus of traditional SOA is support for on-premise application services. This model is no longer valid in situations where a significant part of enterprise systems is delivered in the form of cloud services with a large number of cloud service providers involved (Joshi et al., 2009). With increasing adoption of cloud computing (Khajeh-Hosseini et al., 2010) the primary role of the IT consumer (end-user) organizations is shifting from on-premise implementation of SOA applications to integration of cloud services and management of a hybrid environment that involves both on-premise and cloud services (Farrell, 2011). Public cloud services are hosted on the provider infrastructure and controlled and managed by external service providers that make the services available to consumers on a pay-per-use basis (Breiter and Behrendt, 2009). Service consumers do not have full control over the data and applications

that are provided as cloud services, and consequently cannot guarantee the level of security and availability that they are typically expected to provide (Breiter and Behrendt, 2009). In response to such concerns, there has been significant interest in hybrid cloud architectures where enterprise applications are partly hosted on-premise, and partly in public cloud environments (Hajjat et al., 2010). These developments impact on SOA SDLC as well as on SOA architectural features and functions, resulting in divergence between service consumer and service provider SDLC cycles. As a direct consequence of providing cloud services to large populations of service consumers, cloud service providers cannot take into account the needs of individual service consumers and their service development schedules. Although the investigation of architectural requirements for cloud computing has been the subject of recent research interest and standardization efforts (Liu et al., 2011), most of the work takes the perspective of service providers, and relatively little research has been done so far on issues that impact service consumers. The detail definition of service consumer SDLC and methodological support for cloud service consumers are still an open research problems. In our previous work we discuss the challenges of managing enterprise applications in the cloud context (Feuerlicht and Tran, 2014a) and describe a consumer-side solution in the form of a Service Consumer Framework (SCF) (Feuerlicht and Tran, 2014b). In this paper our focus is on service consumer SDLC; we specify service consumer SDLC phases and describe architectural components required to support the lifecycle activities. The next section (section 2) is a review of related work, and section 3 is a discussion of the traditional SOA SDLC from a service provider perspective. Section 4 describes service consumer SDLC referring to architectural components that are required to support this SDLC. Section 5 contains our conclusions and proposals for future work.

2. RELATED WORK

The NIST (National Institute of Standards and Technology) Cloud Computing Standards Roadmap, Hogan et al. (2011) separates the role of service provider and service consumer and defines a set of cloud computing system development life cycle activities and functions. The activities of cloud providers include service deployment, service orchestration, cloud service management, security and privacy. Activities of cloud consumers include identifying (suitable) services, requesting services, negotiating service contracts, and deploying services. In early research, Papazoglou (2008) proposes a service development lifecycle that contains a preparation planning phase as and five life cycle phases: analysis and design, construction and testing, provisioning, development, and execution and monitoring. The paper describes a methodology and roadmap to assist service providers and service aggregators in assembling multiparty business processes. In a more recent paper, Papazoglou et al. (2011) propose a change-oriented service life-cycle to address issues that arise with changes that cascade across multiple services. The life-cycle starts with the identification of the need for service change and scoping its extent, and then progresses to a service analysis phase that uses the model of the current state of the services (as-is model) and the to-be service model to perform gap analysis. Following the analysis of the impact of the required changes, decisions are made about how to deal with overlapping and conflicting service functionality. During the final change life-cycle phase new services are aligned, integrated, tested and released into production. Using a simple travel service scenario, Ruz et al. (2011) describe a flexible SOA cloud life cycle using SCA (Service Component Architecture) as a model for managing the life cycle of service-based applications. The life cycle consists of three phases: initial design and deployment, runtime monitoring, and design modification and reconfiguration. Authors also introduce an integrated and open framework for supporting flexible cloud service management based on SOA principles. Gu and Lago (2007) present a three phase stakeholder-driven service life cycle that involves three separate roles: service provider, service consumer and service broker. The main objective of this approach is to decouple the activities of service consumer and service provider across development phases: design time, runtime, and change time. In this stakeholder-driven life cycle model, service provider is responsible for service design, service development and testing, and service consumer is responsible for service orchestration, negotiation and monitoring. Focusing on SDLC of cloud service, Breiter and Behrendt (2009) describe a service lifecycle and study the relationship of managing this life cycle and ITIL. This approach focuses on managing IT functionality as one or more aggregated resources exposed as a cloud service. Another approach to managing an integrated lifecycle of IT services in a cloud environment is proposed by Joshi et al. (2009). Authors propose cloud service lifecycle that consist of five sequential phases: requirements, discovery, negotiation, composition, and consumption. The authors have identified performance metrics

associated with each phase: data quality, cost, security, service gap, SLA (Service Level Agreement), QoS (Quality of Service), consumer satisfaction, etc. Pot'vin, et al. (Pot'vin et al., 2013) present a cloud service life cycle that aims to deliver greater adaptability for dynamic business needs, significant operational efficiencies, and lower overall costs. The authors argue that to achieve the best value from the cloud infrastructure, it is important to have end-to-end management and automated workflows for various activities during all the phases of the cloud life cycle; starting with planning and setting up the cloud, to end-user self-service provisioning and de-provisioning of applications, to metering and charging back for the cloud resource usage. While the authors acknowledge that there are multiple stakeholders related to the service lifecycle, they apply Oracle Enterprise Manager 12c Cloud Control (EMC12c) as integrated business and software service lifecycle framework that treats cloud SDLC from the provider perspective. To support the full lifecycle from request to retirement, Farrell (2011) introduces a cloud lifecycle management that aims to ensure successful use of the cloud by implementing policy-driven provisioning processes through a self-service portal supported by a service catalogue. The cloud lifecycle starts with service request submitted by a consumer using self-service portal, and then goes through three different steps: orchestration and provisioning, operations and governance, and service retirement.

Most of the research work on cloud service lifecycle reviewed in this section focuses on provider-side SDLC and proposes methods and procedures that aim to assist providers (or service brokers) to successfully deliver cloud services to consumers. This neglects to address important issues that service consumers face in cloud computing environments. Our focus in this paper is on service consumer SDLC that has its own distinct phases and characteristics that substantially diverge from provider service SDLC. By explicitly identifying the differences between service provider and consumer SDLC cycles, and describing the consumer SDLC phases we aim to develop a lifecycle methodology to support cloud service consumers.

3. SERVICE PROVIDER SDLC

Before we discuss the specifics of service consumer SDLC in the next section (section 4), we briefly review service provider SDLC in this section. While there are some differences in various approaches, most researchers include the lifecycle phases illustrated in Figure 1. The figure represents an iterative and incremental lifecycle process that includes a preparatory *Planning* phase and five distinct main phases: *Analysis and Design*, *Implementation*, *Provisioning*, *Deployment*, and *Execution and Monitoring* (Papazoglou, 2008, Papazoglou and Heuvel, 2007).

The planning phase is a preparatory phase that analyses the business case for different combinations of development approaches and realization strategies. This business case analysis includes gap analysis, and risk analysis. Gap analysis compares planned services with available software services that can be assembled to implement the new business processes. Gap analysis matches high-level descriptions of new services that make up a business process against the available services and includes scenario analysis that considers costs, risks, benefits, and ROI (Return on Investment) associated with the development of new business processes. Risk analysis compares costs and benefits in provisioning scenarios and verifies the feasibility of specific options. The impact and probability of events that may influence the performance of services is estimated. Finally, ROI is calculated for each option based on net benefits and costs.

The analysis and design phase aims to identify and conceptualize business processes as a set of interacting services. Service analysis captures all activities required for the identification and contextualisation of a service, and helps to prioritize

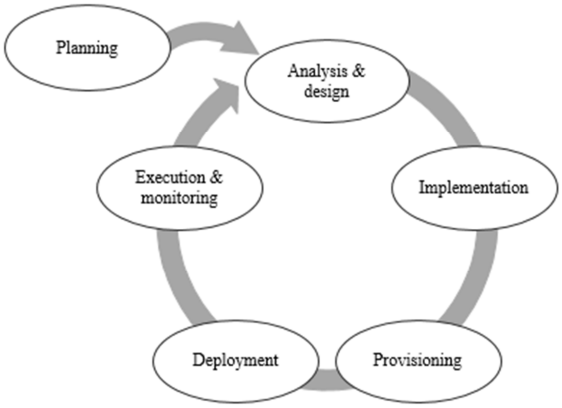

Figure 1: Service Provider SDLC

business processes that offer potential improvements in business value. Services are scoped aggregating simple services and decomposing complex services to reflect business requirements. This step is followed by service design, during which the technical details of service interfaces are specified using design principles that include minimization of coupling and maximization of service cohesion. Importantly, this phase involves decisions about

service granularity and grouping of service operations ensuring that the overlap between the functionality of different services is minimized. The emphasis is on maximising the potential for reuse of resulting services in different application contexts. The design activities are dependent on the specific type of service and may vary according to a delivery strategy.

Service design forms the input for the implementation phase and includes all activities that are related to the realisation of the services based on the detail design developed during the previous phase. As services are reused in different application scenarios services need to be exhaustively tested before they can be published in a service repository. Testing involves ensuring that requirements as specified in previous SDLC phase have been met and that the deliverables are of acceptable quality and conform to the relevant industry standards (Kohlborn et al., 2009). Service provisioning phase involves implementing service governance, certification, auditing, metering, and billing, and controlling the behaviour of services during their use.

Service provisioning can be local or over a network and can involve a complex mixture of technical and business aspects that support various client activities (Farrell, 2011).

The deployment phase involves registering services in various service marketplaces and repositories, determination of access rights, pricing models, and specifying details of the corresponding SLAs.

Service execution and monitoring phase covers activities during service runtime when services are operational, and their progress can be monitored. This phase includes outage management with the objective of maximizing availability of services. Some methodologies (e.g. ITIL) include an additional service improvement phase that involves service revisions and result in the creation of new and improved service versions.

4. SERVICE CONSUMER SDLC

As noted earlier, traditional SOA does not explicitly differentiate between service provider and service consumer SDLC cycles, and assumes that the services are designed, developed and provisioned by the same organization that deploys the services in its on-premise enterprise applications. This assumption is valid in most traditional SOA environments, but does not hold in situation where enterprise applications consume a large number of cloud services provided by external providers with autonomous service lifecycles. 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 reliable operation of services they provide, while service consumers are primarily responsible for the selection of suitable services, integration of cloud services into their enterprise applications, and ensuring continuity of operation at runtime. We have identified the following five phases of service consumer SDLC: *Requirements Specification*, *Service Identification*, *Service Integration*, *Service Monitoring*, and *Service Optimization* (Figure 2). We classify these phases 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 service consumer SDLC is closely interrelated with the Service Consumer Framework (SCF) that provides support for lifecycle phases and activities and consist of Service Repository, Workflow Engine, Service Adaptors, and a Notification Centre (Feuerlicht and Tran, 2014b).

During the service requirements specification phase the service consumer describes 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-

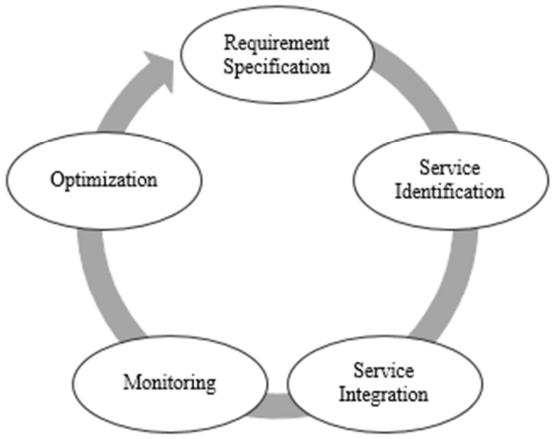

Figure 2: Service Consumer SDLC

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 may create a Request for Service (RFS) and record the information in the service repository (Joshi et al., 2009).

Service Identification (discovery) is constrained by the functional and non-functional requirements documented in the previous phase (requirements specification phase). Service identification phase begins by searching the service consumer repository to attempt to match the service requirements with services that are already recorded in the repository and certified for use. If no existing service matches the requirements, the service consumer will search the service repositories of cloud service providers, or contact a preferred service provider directly to locate a suitable cloud service. Service selection involves identification of suitable cloud services, their testing and approval. Service approval is an internal certification 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 the 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.

Following service identification phase cloud services need to be integrated into consumer enterprise applications. This process consists of the registration of the application and design of workflows using certified services already available in the repository. Workflow design involves searching the service repository for approved services that match the requirements of enterprise applications and composing workflows that control the service execution sequence at runtime. Designers match the desired QoS attributes values against information stored in the repository and define the processing rules that determine the sequence of service execution at run-time (Feuerlicht and Tran, 2014b). This phase involves measuring QoS attributes and comparing their values with those specified in the corresponding SLAs. 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 SCF incorporates a Notification Centre that records service status of cloud services in a runtime log and is used by application administrators to monitor service utilization, planned maintenance activities, and to perform statistical analysis of response times and throughput for individual cloud services.

Service optimization phase is concerned with continuous service improvement. This can be done by replacing existing services with new versions as these become available, or by identifying alternative cloud services from a different provider with identical functionality. For example, the payment service PayPal could be replaced by the SecurePay service. Using service utilization data recorded in the SCF log and information available from cloud service providers (or service brokers) the consumer makes decisions about replacing existing services based on cost and performance, and other relevant QoS parameters. SCF supports this 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 corresponding workflows.

5. CONCLUSIONS

In this paper we have argued that the recent shift towards large-scale use of externally provided public cloud services in enterprise applications has altered the basic premise of SOA, and that this necessitates re-assessment of the SOA paradigm. In particular, the assumption that the life cycle of enterprise services is controlled by the end-user organization (service consumer) is no longer valid. The traditional SOA model that focuses on on-premise application services is no longer sustainable in situations where a significant part of enterprise infrastructure and applications is delivered in the form of cloud services with a large number of cloud service providers involved. In our previous work we have described a proposal for consumer-side solution in the form of a Service Consumer Framework (SCF) that proposes architectural extensions designed to support operation in cloud computing environments. In this paper we have identified the differences between service provider and service consumer SDLC cycles, and we have provided high-level description of the service consumer SDLC. Our future efforts will focus on refining the task descriptions within the various service consumer SDLC phases, and developing a set of corresponding principles and guidelines that

will provide methodological support for cloud service consumers.

REFERENCES

BREITER, G. & BEHRENDT, M. 2009. Life cycle and characteristics of services in the world of cloud computing. *IBM Journal of Research and Development*, 53**,** 3:1-3:8.

FARRELL, K. 2011. *Cloud Lifecycle Management: Managing Cloud Services from Request to Retirement* [Online]. BMC Software. Available: http://www.bmc.com/blogs/hybrid-cloud-delivery-managing-cloud-services-from-request-to-retirement/ [Accessed 03/02/2015 2015].

FEUERLICHT, G. & TRAN, H. T. Enterprise Application Management in Cloud Computing Context. The 8th International Conference on Research and Pratical Issues of Enterprise Information Systems (CONFENIS), 2014a Ha Noi, Vietnam. ACM.

FEUERLICHT, G. & TRAN, H. T. Service consumer framework: Managing Service Evolution from a Consumer Perspective. ICEIS-2014. 16th International Conference on Enterprise Information Systems, 23-30/04/2014 2014b Portugal. Springer.

GU, Q. & LAGO, P. A stakeholder-driven service life cycle model for SOA. 2nd international workshop on Service oriented software engineering: in conjunction with the 6th ESEC/FSE joint meeting, 2007. ACM, 1-7.

HAJJAT, M., SUN, X., SUNG, Y.-W. E., MALTZ, D., RAO, S., SRIPANIDKULCHAI, K. & TAWARMALANI, M. 2010. Cloudward bound: planning for beneficial migration of enterprise applications to the cloud. *SIGCOMM Comput. Commun. Rev.*, 41**,** 243-254.

HOGAN, M., LIU, F., SOKOL, A. & TONG, J. 2011. NIST cloud computing standards roadmap. *NIST Special Publication*, 35.

JOSHI, K., FININ, T. & YESHA, Y. Integrated lifecycle of IT services in a cloud environment. Proceedings of The Third International Conference on the Virtual Computing Initiative (ICVCI 2009), Research Triangle Park, NC, 2009.

KHAJEH-HOSSEINI, A., GREENWOOD, D. & SOMMERVILLE, I. Cloud migration: A case study of migrating an enterprise it system to iaas. Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on, 2010. IEEE, 450-457.

KOHLBORN, T., KORTHAUS, A. & ROSEMANN, M. Business and Software Service Lifecycle Management. Enterprise Distributed Object Computing Conference, 2009. EDOC '09. IEEE International, 1-4 Sept. 2009 2009. 87-96.

LIU, F., TONG, J., MAO, J., BOHN, R., MESSINA, J., BADGER, L. & LEAF, D. 2011. NIST cloud computing reference architecture. *NIST Special Publication*, 500**,** 292.

PAPAZOGLOU, M. 2008. Web Services Development Lifecycle. *Web services: principles and technology*. Pearson Education.

PAPAZOGLOU, M. P., ANDRIKOPOULOS, V. & BENBERNOU, S. 2011. Managing Evolving Services. *IEEE Software*, 28**,** 49-55.

PAPAZOGLOU, M. P. & HEUVEL, W.-J. V. D. 2007. Business process development life cycle methodology. *Commun. ACM*, 50**,** 79-85.

POT'VIN, K., AKELA, A., ATIL, G., CURTIS, B., GORBACHEV, A., LITCHFIELD, N., NELSON, L. & SHARMAN, P. 2013. Cloud Lifecycle Management. *Expert Oracle Enterprise Manager 12c*. Apress.

RUZ, C., BAUDE, F., SAUVAN, B., MOS, A. & BOULZE, A. Flexible SOA Lifecycle on the Cloud Using SCA. Enterprise Distributed Object Computing Conference Workshops (EDOCW), 2011 15th IEEE International, Aug. 29 2011-Sept. 2 2011 2011. 275-282.