Service Development Life Cycle for Hybrid Cloud Environments

Hong Thai Tran[1, 2*], George Feuerlicht[1, 2]

[1] Faculty of Engineering and Information Technology, University of Technology, Sydney, Australia.
[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.

* Corresponding author. Tel.: +61415793935; email: HongThai.Tran@uts.edu.au
Manuscript submitted March 10, 2016; accepted April 5, 2016.
doi: 10.17706/jsw.11.7.704-711

Abstract: With increasing adoption of cloud computing there is a need to provide methodological and tool support for the development of enterprise applications that utilize cloud services. Traditional approaches that assume that services are developed and deployed on-premise are not suitable for hybrid cloud environments, where a significant part of enterprise applications is delivered in the form of cloud services provided by autonomous cloud providers. In this paper we describe a Service Development Life Cycle for hybrid cloud environments and a prototype system designed to support this life cycle.

Key words: Cloud Computing, Service-Oriented Architecture, Service Development Life Cycle, Hybrid Cloud

1. Introduction

Today, enterprise applications typically involve the use of both on-premise and cloud services resulting in hybrid cloud environments [1], [2]. According to Gartner Special Report on the Outlook for Cloud [3], half of large enterprises will adopt and use hybrid cloud model by the end of 2017. As a result of the extensive use of cloud services, end-user organizations no longer control the entire service System Development Life Cycle (SDLC) and rely on cloud service providers to ensure the quality and availability of enterprise applications. The traditional service SDLC assumes that services are designed, implemented, provisioned and deployed on-premise, and that end-user organizations manage the operation and evolution of services. This approach is no longer suitable in situations that involve hybrid clouds where a significant part of enterprise applications is delivered in the form of cloud services by cloud service providers [4]. The traditional service SDLC needs to be extended to address issues that include the identification, monitoring and management of cloud services. The life cycle must support the selection of cloud services that satisfy application requirements from a large range of services offered by various providers with different cost models and QoS (Quality of Service) attributes. Another area that needs to be addressed includes monitoring and management of cloud services at runtime to maintain operational continuity.

In our earlier work [5], [6], we argued that the use of cloud services in enterprise applications necessitates re-assessment of the SOA paradigm, and in particular the service SDLC. We have identified differences between service provider and service consumer SDLC cycles and described the service consumer SDLC phases in detail. We have also proposed a design of service repository to support the information requirements of the various SDLC phases. In this paper, we adapt the SDLC to hybrid cloud environments and present a prototype tool designed to support this SDLC. In the next section (Section 2) we review related research on service life cycle management. Section 3 presents the proposed SDLC for hybrid cloud environments and Section 4 describes the prototype tool designed to support the hybrid cloud service SDLC. The final section (Section 5) are our conclusions and proposals for future work.

2. Related Work

Cloud system development is an active research topic today. To encourage the improvement of cloud enterprise system, Schmidt [7] proposed a cloud enterprise system lifecycle which contains of five phases: Service Integration, Resource Import, Service and Resource Configuration, Operation, and Service Disintegration. The author discusses the notion of flexibility and configurability of cloud systems and the idea of using business process management. Using case study of Taiwanese government cloud information system, Kao, et al. [8] provide a framework that supports a Secure System Development Life Cycle (SSDLC) and security enhancement model for cloud applications. The Cloud SSDLC contains five phases: Initiation, Development, Implementation, Operation, and Destruction. Breiter and Behrendt [9] present a five-phase life cycle (Definition, Offering, Subscription and Instantiation, Production and Termination) and discuss the relationship between this life cycle methodology and the Information Technology Infrastructure Library (ITIL) practices. The approach focuses on managing IT functionality as one or more aggregated resources externalized as cloud services. Joshi, et al. [10] describe a cloud based integrated life cycle for cloud services distributed over Internet. This service life cycle is divided into five main phases: Requirements, Discovery, Negotiation, Composition, and Consumption. Schneider and Sunyaev [11] develop a life cycle framework named CloudLive which combines both cloud computing and inter-organizational characteristics. The life cycle has six phases (Requirements Determination, Development Acquisition, Integration, Contract Fulfillment, and Retirement) that involve both provider and customer perspectives. Ruz, et al. [12] describe a flexible SOA cloud life cycle using the Service Component Architecture. Gu and Lago [13] present a three phase stakeholder-driven service life cycle. Pot'vin, *et al*. [14] present a cloud service life cycle that aims to deliver greater adaptability for dynamic business needs, significant operational efficiencies, and lower overall costs. Finally, Kohlborn, *et al*. [15] present a generic business and software service life cycle and align it with the common management layers within the organisation.

Most of the research dealing with the cloud service life cycle reviewed in this section focuses on provider-side SDLC and proposes methods and procedures that aim to assist service providers (or service brokers) to successfully deliver cloud services to consumers, and do not address important issues that service consumers face in hybrid cloud environments. Our focus is on service SDLC for hybrid cloud environments that has distinct phases and characteristics that substantially diverge from the traditional SDLC for on-premise services. We explicitly identify the differences between service provider and consumer SDLC cycles, describe the SDLC phases for hybrid cloud environments, and propose a methodology for the development of hybrid cloud applications.

3. Service Development Life Cycle for Hybrid Cloud Environments

In traditional SOA environments the system development life cycle focuses on the implementation of on-premise services. In cloud context, enterprise applications consume externally provided cloud services that are the subject of autonomous service provider life cycles with service providers responsible for the implementation and reliable operation of services. Service consumers are primarily responsible for the selection of suitable services, integration of cloud services into their enterprise applications, and ensuring the continuity of operation at runtime. We have identified the following five phases of the SDLC for hybrid cloud environments: Requirements Specification, Service Identification, Service Integration, Service

Monitoring, and Service Optimization (Fig. 1). We describe these life cycle phase in detail in the following sections.

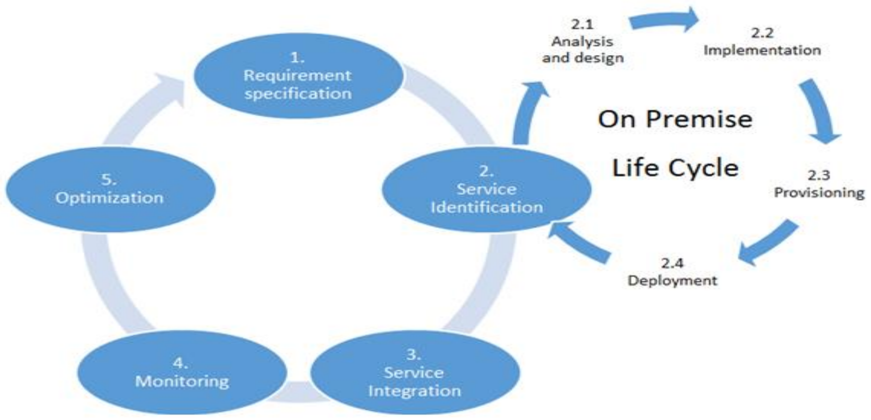


Fig. 1. Hybrid service consumer SDLC.

3.1. Requirements Specification

The Requirement Specification phase (phase 1) describes the functional and non-functional requirements that a given service needs to fulfill and serves as a reference for judging available design alternatives. A software requirement specification is the first detailed documentation of the desired behavior of a software system. Errors introduced during the requirements analysis phase are difficult and expensive to correct once they are propagated into the design and implementation phases, and can have a major impact on the functionality and reliability of the resulting system [2]. Once a service is fully described and classified, the information is stored in a service repository and the service consumer creates a Request for Service (RFS) that contains all functional (service operations, inputs and output parameters, etc.) and non-functional requirements (throughput, response time, availability, etc.) that the software developer needs to design and implement the service. It is important that non-functional parameters are measurable so that candidate cloud services can be matched against the requirements (e.g. the throughput requirement could be expressed as the percentage, e.g. 95% of transactions processed in less than 10 seconds). The QoS requirements are used in later SDLC stages to identify a suitable design and play an important role in determining the efficiency and adaptability of the system.

3.2. Service Identification

Service Identification (phase 2) is constrained by the functional and non-functional requirements documented in the requirements specification phase. Firstly, the functional requirements of candidate services are matched against the RFS, and the nonfunctional requirements (i.e. availability, response time, security, cost, etc.) are considered for each candidate service. The service identification phase starts by attempting to identify suitable services in the enterprise service repository (i.e. already certified and documented services). If the service repository does not contain any suitable services, then the search continues by looking for cloud services that match the RFS specification. Finally, if no candidate cloud services match the requirements, the service will need to be developed and deployed on-premise following the On-Premise SDLC cycle illustrated in Fig. 1.

The *Analysis and Design* phase (phase 2.1) aims to identify and conceptualize business processes as a set of interacting services. Service analysis captures all activities required for the identification and contextualization of a service, and helps to prioritize business processes that offer potential improvements in business value. The design activities are dependent on the specific type of service and may vary according to a service delivery strategy.

Service design forms the input for the *Implementation* phase (phase 2.2) that includes activities related to the realization of the services based on the detail design developed during the previous phase. Services need to be exhaustively tested before they can be published in the service repository. Testing involves ensuring that requirements as specified in the previous SDLC phase have been met and that the deliverables are of acceptable quality and conform to the relevant industry standards [15].

Service *Provisioning phase* (phase 2.3) involves implementing service governance, certification, auditing, metering, and billing, and controlling the behavior of services during execution. 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 [1]

Service *Deployment phase* (phase 2.4) involves registering services in the service repository, determination of access rights, pricing models, and specifying details of the corresponding SLAs (Service Level Agreements).

3.3. Service Integration

The main SDLC cycle continues with the *Integration* phase (phase 3). This phase involves the design of workflows using certified services (i.e. services already certified and stored in the service repository). Workflow design involves identifying services that match the requirements of enterprise applications and composing workflows that control the service execution sequence at runtime. There are two main activities in the service integration phase: service consumption and workflow integration. To integrate a service into the enterprise system, the service consumer needs to review the SLA that defines service interfaces, delivery mode, quality metrics, security and cost of the service, and then integrate the service using a specific technological platform, ensuring reliability and providing facilities for service management.

3.4. Service Monitoring

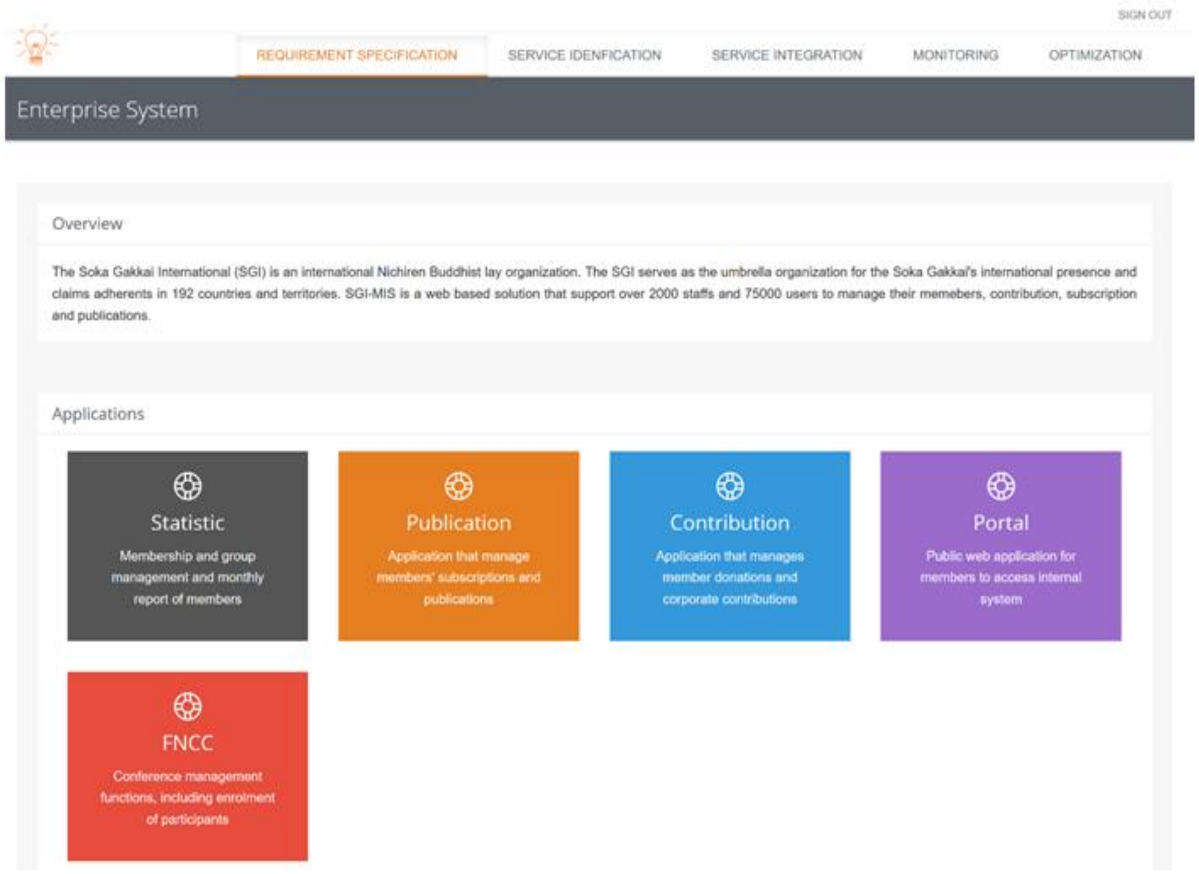
Service *Monitoring* phase (phase 4) involves collecting data required to manage and control the behavior of services during runtime. Monitoring the service behavior is very important in order to maintain the service performance, validation, and integrity. Monitoring data, including response time, results of service invocations (i.e. error, invalid or success states) and response messages, is stored in service logs. Service logs are analyzed to calculate the actual performance attributes of the services such as availability and response time. Using these performance attributes, enterprise system performance can be measured and compared against the requirements specification developed during early SDLC phases and used for reporting of defects and management of system reliability and continuity.

3.5. Service Optimization

The final service *Optimization* phase (phase 5) is concerned with continuous service improvement. This can be done by replacing existing services with new service versions as these become available, or by identifying an alternative cloud service from a different provider with identical functionality. Alternatively, business processes can change to optimize overall system performance. At the technological level service workflows can be re-configured in order to improve system performance or to reduce the overall cost. For example, in the Payment Workflow, the payment service PayPal could be replaced by the SecurePay service to improve system availability. Using service utilization data recorded during the monitoring phase, the service consumer can make decisions about replacing existing services based on cost and performance, and other relevant QoS parameters.

4. Prototype Implementation

We are developing a prototype system to support the service life cycle for hybrid cloud described in the previous section. The prototype is developed using ASP.Net Web technologies to build the front end of the SCF framework (Service Consumer Framework), Microsoft SQL Server database implemented as AWS (Amazon Web Services) RDS (Relational Database Service), and Entity Framework as an object-relational modeling technology. We use the prototype system to design and implement a conference management application that contains five sub-systems: Contributions, Subscription, Statistic, Conference Management and an Online Portal. In addition to on-premise services, the applications are utilizing numerous cloud services, including payment services, storage services, and travel services. The prototype platform supports hybrid service life cycle described in the previous section; the menu options on the top of the screen in Fig. 2 enable input of information for the individual SDLC phases.



Overview

The Soka Gakkai International (SGI) is an international Nichiren Buddhist lay organization. The SGI serves as the umbrella organization for the Soka Gakkai's international presence and claims adherents in 192 countries and territories. SGI-MIS is a web based solution that support over 2000 staffs and 75000 users to manage their memebers, contribution, subscription and publications.

Applications

Statistic
Membership and group management and monthly report of members

Publication
Application that manage members' subscriptions and publications

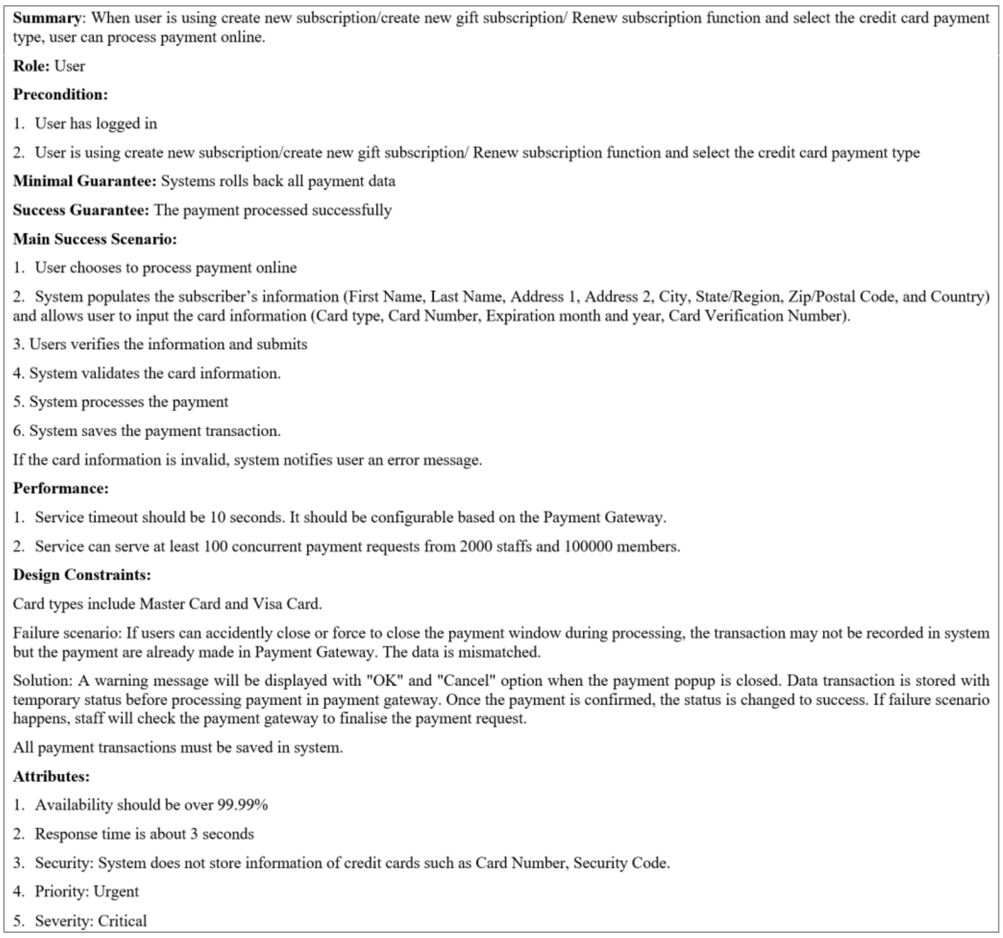
Contribution
Application that manages member donations and corporate contributions

Portal
Public web application for members to access internal system

FNCC
Conference management functions, including enrolment of participants

Fig. 2. User interface for service life cycle management.

During the requirements analysis phase we specify the enterprise applications using a number of RFS specifications. An example of Online Payment RFS for the Subscription subsystem is shown in Fig. 3. Using this information we can search the service repository matching functional and non-functional requirements. Cloud services are divided into different service categories or service domains: Security, Payment, Storage and Travel. The search function queries the service repository matching QoS attributes such as availability, response time, cost and capacity to identify suitable candidate services. The QoS attributes are computed from data collected during the monitoring phase. Following the identification of services, workflows are configured to implement the desired business functionality and to maximize availability and performance using redundant replacement services. During the service monitoring phase the system continuously analyses information recorded in service logs and notifies application administrators about service outages and reduced availability. This information is also used to support service optimization at run-time and to ensure that the nonfunctional requirements (e.g. availability, response time, etc.) meet the requirements specification. For example, the monitoring data may indicate that the availability of the Online Payment service is 99.5%; significantly less than 99.9% as specified in the RFS. The SCF framework will notify system administrators and this may lead to reconfiguration of the payment workflow, or the replacement of the payment service by another service with better performance characteristics.



Summary: When user is using create new subscription/create new gift subscription/ Renew subscription function and select the credit card payment type, user can process payment online.

Role: User

Precondition:

1. User has logged in
2. User is using create new subscription/create new gift subscription/ Renew subscription function and select the credit card payment type

Minimal Guarantee: Systems rolls back all payment data

Success Guarantee: The payment processed successfully

Main Success Scenario:

1. User chooses to process payment online
2. System populates the subscriber's information (First Name, Last Name, Address 1, Address 2, City, State/Region, Zip/Postal Code, and Country) and allows user to input the card information (Card type, Card Number, Expiration month and year, Card Verification Number).
3. Users verifies the information and submits
4. System validates the card information.
5. System processes the payment
6. System saves the payment transaction.

If the card information is invalid, system notifies user an error message.

Performance:

1. Service timeout should be 10 seconds. It should be configurable based on the Payment Gateway.
2. Service can serve at least 100 concurrent payment requests from 2000 staffs and 100000 members.

Design Constraints:

Card types include Master Card and Visa Card.

Failure scenario: If users can accidently close or force to close the payment window during processing, the transaction may not be recorded in system but the payment are already made in Payment Gateway. The data is mismatched.

Solution: A warning message will be displayed with "OK" and "Cancel" option when the payment popup is closed. Data transaction is stored with temporary status before processing payment in payment gateway. Once the payment is confirmed, the status is changed to success. If failure scenario happens, staff will check the payment gateway to finalise the payment request.

All payment transactions must be saved in system.

Attributes:

1. Availability should be over 99.99%
2. Response time is about 3 seconds
3. Security: System does not store information of credit cards such as Card Number, Security Code.
4. Priority: Urgent
5. Severity: Critical

Fig. 3. Online payment request for service.

5. Conclusion

In this paper we have described a system development life cycle for hybrid cloud environments suitable for enterprise systems that use both cloud and on-premise services. This SDLC life cycle has been adapted from the traditional SOA life cycle to take account of the situation where a significant part of enterprise infrastructure and applications is delivered in the form of cloud services. The hybrid SDLC consists of five main phases (Requirements Specification, Service Identification, Service Integration, Service Monitoring, and Service Optimization) and a related on-premise life cycle for services that are not available as cloud services and need to be developed on-premise. We have also described a prototype of the SCF framework that supports the various SDLC phases. We are currently enhancing the SCF framework to support all SDLC activities throughout the entire hybrid SDLC cycle. Our future efforts will focus on refining SDLC activities and developing a set of corresponding principles and guidelines that will provide a comprehensive methodological support for the entire hybrid service SDLC.

References

[1] Farrell, K. *Cloud Lifecycle Management: Managing Cloud Services from Request to Retirement*. Retrieved March 2, 2011, from http://www.bmc.com/blogs/hybrid-cloud-delivery-managing-cloud-services-from-request-to-retirement

[2] Hajjat, M., Sun, X., Sung, Y.-W. E., Maltz, D., Rao, S., Sripanidkulchai, K*., et al.*, (2010). Cloudward bound: Planning for beneficial migration of enterprise applications to the cloud, *SIGCOMM Comput. Commun. Rev., 41*.

[3] Janessa, R., & Meulen, R. V. d. *Gartner Special Report Examines the Outlook for Hybrid Cloud*. Retrieved October 1, 2013, from http://www.gartner.com/newsroom/id/2599315

[4] Joshi, K., Finin, T., & Yesha, Y. (2009). Integrated lifecycle of IT services in a cloud environment, *Proceedings of the 3rd International Conference on the Virtual Computing Initiative*.

[5] Feuerlicht, G., & Tran, H. T., (2015). Adapting service development life-cycle for cloud. *Proceedings of the 17th International Conference on Enterprise Information Systems*.

[6] Tran, H. T., & Feuerlicht, G. (2015). Service repository for cloud service consumer life cycle management. *Service Oriented and Cloud Computing*, 171-180.

[7] Schmidt, R., (2012). Conceptualisation and lifecycle of cloud based information systems. *Proceedings of the 2012 IEEE 16th International,Enterprise Distributed Object Computing Conference Workshops.* (pp. 104-113).

[8] Kao, T. C., Mao, C. H., Chang, C. Y., & Chang, K. C. (2012). Cloud SSDLC: Cloud security governance deployment framework in secure system development life cycle. *Proceedings of the 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)* (pp. 1143-1148).

[9] 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.

[10] Joshi, K. P., Yesha, Y., & Finin, T. (2014). Automating cloud services life cycle through semantic technologies. *IEEE Transactions on Services Computing, 7*, 109-122.

[11] Schneider, S., & Sunyaev, A. (2015). CloudLive: a life cycle framework for cloud services, *Electronic Markets, 2*, 299-311.

[12] Ruz, C., Baude, F., Sauvan, B., Mos, A., & Boulze, A. (2011). Flexible SOA lifecycle on the cloud using SCA. *Proceedings of the 2011 15th IEEE International Enterprise Distributed Object Computing Conference Workshops* (pp. 275-282).

[13] Gu, Q., & Lago, P. (2007). A stakeholder-driven service life cycle model for SOA. *Proceedings of the 2nd International Workshop on Service Oriented Software Engineering: In Conjunction with the 6th ESEC/FSE Joint Meeting* (pp. 1-7).

[14] Pot'vin, K., Akela, A., Atil, G., Curtis, B., Gorbachev, A., & Litchfield N*., et al.*, (2013). Cloud lifecycle management. *Expert Oracle Enterprise Manager*.

[15] Kohlborn, T., Korthaus, A., & Rosemann, M. (2009). Business and software service lifecycle management. *Proceedings of the IEEE International Enterprise Distributed Object Computing Conference* (pp. 87-96).

Hong Thai Tran is currently PhD student of Faculty of Engineering and Information Technology, University of Technology, Sydney, Australia. He has received his master of science in internetworking from University of Technology, Sydney in 2008. He is currently a lecturer of Business Information System Faculty, University of Economics, Ho Chi Minh city, Vietnam. He has over 15 years of research and teaching experience as well as working experience in IT industry.

His research interests are in the areas of service oriented architecture, distributed systems, enterprise architecture and cloud computing.

George Feuerlicht has received his PhD in electrical engineering from Imperial College, London University, UK in 1980. He is an expert in the area of database management and has over 15 years of research, consulting, and teaching experience with the application of database management, enterprise architecture and cloud computing. George is currently the director of a master course in professional computing course at the Faculty of Engineering and Information Technology, University of Technology, Sydney, Australia. He is a member of ACM, AIS, the Australian section of the ISO Working Group on database language standards and several international program committees and editorial boards. He is the author of over 70 publications across a range of topics in information systems and computer science, including recent publications on enterprise architectures, SOA, and Cloud Computing models.