SERVICE-BASED APPLICATION DEVELOPMENT ON THE CLOUD
*State of the Art and Shortcomings Analysis**

Dinh Khoa Nguyen, Yehia Taher, Mike P. Papazoglou and Willem-Jan van den Heuvel
European Research Institute in Service Science (ERISS), Tilburg, The Netherlands

Keywords: Cloud Application, SaaS, PaaS, IaaS, State-of-the-Art.

Abstract: Recently, Cloud Computing has become an emerging research topic in response to the shift from product-oriented economy to service-oriented economy and the move from focusing on software/system development to addressing business-IT alignment. From IT perspectives, there is a proliferation of methods for cloud application development. Such methods have clearly shown considerable shortcomings to provide an efficient solution to deal with major aspects related to cloud applications. One of these major aspects is the multi-tenancy of the Software-as-a-Service (SaaS) components used to compose Service-Based Applications (SBAs) on the cloud. Current SaaS offerings are often provided as monolithic *one-size-fits-all* solutions and give little or no opportunity for further customization. As a result, monolithic SaaS offerings are more likely to show failure in meeting the business requirements of several consumers. In this paper, we analyze the state-of-the-art of the standardization and methodology support for SBA development on the cloud, identify some shortcomings, and point out the need of a novel approach for breaking down the monolithic stack of cloud service offerings and providing an effective and flexible solution for SBA designers to select, customize, and aggregate cloud service offerings coming from different providers (Nguyen et al., 2011).

1 INTRODUCTION

Service-Oriented-Architecture (*SOA*) (Papazoglou and van den Heuvel, 2006) is a philosophy of design that can be informally described as "the software equivalent of Lego bricks" where a collection of mix-and-match units (called "services") - each performing a well-defined task - can reside on different machines possibly under the control of a different service provider, and are ready to be used whenever needed. Enterprises typically use a single software service to accomplish a specific business task, such as billing or inventory control or they may compose several software services to create a value-added distributed service-based application (SBA) such as customized ordering, customer support, procurement, and logistical support.

However, a serious limitation of SOA is that it does not make any assumptions regarding service deployment and it leaves it up to the discretion of the service developer to make this deployment choice, which is a daunting task and often leads to failure. A dangerous "*difficult-to-customize, one-size-fits-all*" philosophy permeates SOA development leading to brittle implementations where once an application is deployed it is bound to a particular infrastructure. In addition, traditional SOA software development concentrates on a kind of "*big design upfront*" where the prevailing belief is that it is possible to gather all of a developer's or customer's requirements, upfront, prior to coding a software solution. So despite its promises SOA has so far failed to deliver promised benefits except in rare situations leading yet again to a software development crisis.

To address these serious shortcomings it is normal to turn our attention to *Cloud Computing* as it aim to provide both the economies of scale of a shared infrastructure as well as a flexible delivery model that naturally complements the service orientation of SOA. Cloud computing is a computing model for enabling convenient and on-demand network access to a shared pool of configurable and often virtualised computing resources (e.g., networks, servers, storage, middleware and applications as services) that can be rapidly provisioned and released with minimal management effort or service provider interaction (Armbrust et al., 2009). Cloud capabilities are defined and

*The research leading to this result has received funding from the Dutch Jacquard program on Software Engineering Research via contract 638.001.206 SAPIENSA; and the European Union's Seventh Framework Programme FP7/2007-2013 (4CaaSt) under grant agreement n^o 258862.

Nguyen D., Taher Y., P. Papazoglou M. and van den Heuvel W..
SERVICE-BASED APPLICATION DEVELOPMENT ON THE CLOUD - State of the Art and Shortcomings Analysis.
DOI: 10.5220/0003902103950400
In *Proceedings of the 2nd International Conference on Cloud Computing and Services Science* (CLOSER-2012), pages 395-400
ISBN: 978-989-8565-05-1

provided as services where users of cloud-related services are able to focus on what the service provides them rather than how the services are implemented or hosted. This begins to explain why the service orientation provided by SOA needs the "cloud" as a natural deployment medium. In fact, the two concepts can be paired to support service development and deployment and their merger can provide complete services-based solutions. Cloud computing is typically divided into three levels of hosting service offerings: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). These levels support the virtualisation and management of different levels of the computing solution stack.

Having explained the benefit of our "SOA meets cloud" vision, our contribution in this paper is an extensive state-of-the-art analysis on the current standardization and methodology support for SBA development on the cloud, which then leads to identifying some shortcomings of the current support. The rest of the paper is organized as follows. Section 2 presents an extensive state-of-the-art evaluation. Section 3 identifies shortcomings of existing support, and then highlights some research questions and challenges derived from the evaluation of the state-of-the-art. Finally, Section 4 concludes the paper.

2 STATE-OF-THE-ART EVALUATION

Current SBA application development on the cloud usually leads to a vendor lock-in approach, where the constituting monolithic SaaS components are predominantly tethered to proprietary platforms and infrastructure of a cloud vendor and thus provide little or no room for customization, extension and composition. That is because SBA developments do not usually put focus on the deployment environment of the constituent SaaS components. This limitation can be addressed by breaking the monolithic SaaS offerings into cloud services (XaaSs) across cloud computing layers, i.e. SaaS, PaaS and IaaS. Following the SOA principles and techniques, SBA developers can reuse and combine distributed cross-layered XaaS functions. A SOA-enabling SBA development on the cloud results in an amalgamation of on-premise and external XaaSs that promotes the reusability and composability of XaaSs across SaaS providers and PaaS/IaaS cloud vendors. However, cloud computing is a relatively new research area and only a few existing work supports our envisioned SBA development methodology. Section 2.1 reviews and evaluates the existing approaches towards a standardized XaaS representation for supporting the platform-agnostic vendor-independent SBA development on the cloud. From the methodological point of view, Section 2.2 presents the related methodologies to develop SBAs independent from the underlying cloud platform and infrastructure.

2.1 Standardization Support for SBA Development on the Cloud

While developing SBA on the cloud, the absence of standardization across cloud vendors, results in unnecessary complexity to obtain interoperability, high switching costs and potential vendor lock-in. The main concerns of cloud-based SBA development are how to deal with the standardization and interoperability between different cloud platforms (Tsai et al., 2010), since cloud computing promises to allow developers to design and develop elastic and inexpensive applications independent of platforms (Armbrust et al., 2009). However, current cloud vendors have different application models, many of which are proprietary, vertically integrated cloud stacks that limit the customizations of the underlying platform and infrastructure resources. There is currently little effort in supporting tools, techniques, procedures or standard data formats or service interfaces that could guarantee data, application and service portability. In (Monteiro et al., 2011) the vendor lock-in problem that prevents the interchangeability and interoperability between the SaaSs has been addressed and subsequently a state-of-the-art in both standardization efforts and on-going projects has been presented. Document (Vambenepe, 2009) points out that concerning the vendor lock-in there are still many unsolved compatibility issues beside the API compatibility, such as the data format, billing, metering, error handling, logging, or cloud management and administration. In general, the current situation makes it difficult for SBA developers to migrate data and service components from one cloud vendor to another or back to an in-house IT environment.

The ability to manipulate, integrate and customize XaaS across different cloud providers for SaaS development has been studied in (Keahey et al., 2009) that has IaaS, application and deployment orchestrators but falls short of proposing a solution for the problem at hand. The DMTF has published standards such as the Open Virtualization Format (OVF) [2] to provide an open packaging and distribution format for virtual machines, and the Virtualization Management (VMAN) [3] specifications that address the

[2] OVF: http://www.dmtf.org/standards/ovf

[3] VMAN: http://dmtf.org/standards/vman

management lifecycle of a virtual environment to help promote interoperable cloud computing service. The OVF is considered nowadays as a standardized means for describing single or multiple virtual machines. Using the OVF allows for specifying either the technical offering of an IaaS provider or the resource requirements of a SaaS or PaaS provider. Similar to OVF-based approaches, the Solution Deployment Descriptor (SDD) template [4] proposed by OASIS defines an XML schema to describe the characteristics of an installable unit (IU) of software that are relevant for core aspects of its deployment, configuration, and maintenance. The benefits of this work include: the ability to describe software solution packages for both single and multi-platform heterogeneous environments, the ability to describe software solution packages independent of the software installation technology or supplier, and the ability to provide information necessary to permit full lifecycle maintenance of software solutions. The work in (Bernstein et al., 2009) targets the interoperability between the federated clouds by providing a collection of proposals for "Inter-cloud" protocols and formats. However, this work is still in the early stage and targets only the interoperability between the data centers, i.e. only on the infrastructure level.To unlock the vendor lock-in problem concerning the APIs, the Open Grid Forum's Open Cloud Computing Interface (OCCI) working group has been developing a uniform API specification for remote management of Cloud Computing infrastructure [5]. This will allow for the development of interoperable tools for common tasks including deployment, autonomic scaling and monitoring. The scope of the specification will be all high-level functionality required for the life-cycle management of virtual machines (or workloads) running on virtualization technologies (or containers) supporting service elasticity.

Approaching the cloud application development from a different perspective, the Model Driven Engineering (MDE) research community has realized the benefit of combining MDE techniques with SaaS development and suggested combining MDE with cloud computing (Brunelière et al., 2010). As the article describes, there is no consensus on the models, languages, model transformations and software processes for the model-driven development of cloud-based SaaSs. Following the MDE vision, (Hamdaqa et al., 2011) proposes a meta-model that allows cloud users to design applications independent of any platform and build inexpensive elastic applications. From their point of view, a SaaS application should avoid the vendor lock-in problem concerning the underlying platforms. This meta-model allows for describing the capabilities, technical interfaces, and configuration data for the virtualized infrastructure resources of the cloud application service. Similarly, (Cai et al., 2009) presents a different customer-centric cloud service model. This model concentrates on aspects such as the customer subscription, capability, billing, etc., yet does not cover other technical aspects of the cloud services including the technical interfaces of the cloud services, the elasticity, the required deployment environment, etc. Other existing models, e.g. (Thrash, 2010), also lack a formal structure and dentitions (reducing their usability and reusability) or are not explicit and assume tacit knowledge.

In practice, an attempt to provide a template-based approach for using cloud services is available from Amazon through their AWS CloudFormation offering [6]. This template provides AWS developers with the ability to specify a collection of AWS cloud resources and the provisioning of these resources in an orderly and predictable fashion. Nevertheless, this template works only for AWS cloud platform and infrastructure resources and thus lacks interoperability.

In summary, existing approaches mostly target the infrastructure levels and covers only certain perspective of standardizations for cloud services, e.g. description format, APIs, protocol, definition models, protocols, SLA, etc. The state of the art analysis has shown that there is a lack of a uniform representation for cross-layered XaaSs that unifies all views on an XaaS, e.g. from the customer view on the APIs and SLA, to the developers that are responsible for deploying and maintaining that XaaS through the cloud.

2.2 Cloud-based SBA Development Methodologies

Apart from the standardization supports for the interoperability and portability between cloud vendors, we are also interested in the existing methodologies that provide guidelines for developing composite SBA applications on the cloud, some of which are developed based on the XaaS description standards mentioned in the previous section 2.1.

La et al. propose in (La and Kim, 2009) a systematic process for developing high-quality cloud SaaSs, taking into considerations the key design criteria for SaaSs and the essential commonality/variability analysis to maximize the reusability. Although this approach claims to develop cloud-based SaaSs, it does

[4]SDD 1.0: http://docs.oasis-open.org/sdd/v1.0/os/dd-spec-v1.0-os.html

[5]OCCI: http://occi-wg.org

[6]AWS CloudFormation, http://aws.amazon.com/de/cloudformation/

not discuss about the cloud support for the deployment environment of the SaaSs. Maximilien et al. introduces in (Maximilien et al., 2009) a cloud-agnostic middleware that can sit on top of many PaaS/IaaS offerings and enable a platform-agnostic SaaS development. They provide a meta-model for describing SaaS applications and their needed cloud resources, and APIs and middleware services for the deployment. The connection between SOA and cloud computing has been established by the Service-Oriented Cloud Computing Architecture (SOCCA) proposed in (Tsai et al., 2010). Using the SOCCA, developers can build SBA applications following an integrated SOA framework. Cloud platform and infrastructure resources will be discovered by a Cloud Broker Layer and a Cloud Ontology Mapping Layer for deploying the SaaS components. The multi-tenancy feature of cloud computing is also supported by SOCCA where multiple instances of SaaS applications or components can be provided to multiple tenants. Although the SOCCA is a useful reference architecture for developing cloud-based SBAs following the SOA paradigm, a lot of supports are lacking here including a concrete definition language for the SaaS components, a mapping approach for finding necessary cloud resources, and the ability to specify and resolve end-to-end constraints of SaaS applications that might affect the underlying cloud resources.

The Cafe application and component templates in (Mietzner, 2010) are a relevant approach for cloud-based SaaS development that provides an ad-hoc composition technique for application components and cloud resources following the Service Component Architecture (SCA). However, this approach requires SaaS developers to possess deep technical knowledge of the application architecture and the physical cloud deployment environment to select and compose the right application components and cloud resources.

(Galán et al., 2009) uses the OVF to define a service definition language for deploying complex SaaS applications in federated IaaS clouds. These SaaS applications consist of a collection of virtual machines (VMs) with several configuration parameters (e.g., hostnames, IP addresses and other application specific details) for software components (e.g., web/application servers, database, operating system) running on the VMs. The service definition language enables also the specification of SaaSs' Key Performance Indicators (KPIs) and the elasticity rules that prescribe what to do in case the KPIs do not meet the expected levels. (Chapman et al., 2010) extends this language into a service definition manifest to serve as a contract between a SaaS provider and the infrastructure provider. In this contract, architectural constraints and invariants regarding the infrastructure resource provisioning for an application service are specified and can be used for on-demand cloud infrastructure provisioning at run-time. KPIs monitoring mechanisms are also specified in the contract for ensuring the timely scaling of the provisioned infrastructure resources based on the specified elasticity rules. Nevertheless, the existing work related to the OVF targets the infrastructure level only, i.e., they allow the specification of architecture constraints for deploying the applications directly on (federated) data centres but do not cover the holistic picture of a top-down development of SBAs on the cloud that can guide developers in selecting, resolving and composing cross-layered XaaS offerings. Using the service definition manifest to specify the structure of a SaaS application, i.e. the SaaS components and their required Virtual Execution Environments (VEE), the Reservoir architecture (Rochwerger and et al., 2009) can automatically provision the VEE instances that can run simultaneously without conflict on a federated cloud infrastructure of multiple providers. KPI monitoring mechanisms and elasticity rules in the manifest act as a contract that guarantees the required Service Level Agreement (SLA) between the SaaS provider and the Reservoir architecture.

Model-driven approaches are also employed for the purpose of automating the deployment of complex IaaS services on cloud infrastructure. For instance, (Konstantinou et al., 2009) propose a virtual appliance model, which treats virtual images as building blocks for IaaS composite solutions. Virtual appliances are composed into virtual solution model and deployment time requirements are then determined in a cloud-independent manner using a parameterized deployment plan. In a similar way, (Chieu et al., 2010) describes a solution-based provisioning mechanism using composite appliances to automate the deployment of complex application services on a cloud infrastructure.

In summary, the state of the art analysis has shown that there is a need for a methodology that guides cloud-based SBA developers to make informed decisions for selecting, customizing, and composing cross-layered XaaS offerings from multiple providers. The methodology should obey the SOA principles and techniques that promote the reusability, loose coupling and composability of the underlying XaaSs.

3 SHORTCOMINGS OF EXISTING APPROACHES

This section summarizes the shortcomings we iden-

Table 1: Summary of existing gaps and innovations needed to address them.

Shortcomings	Research Challenges
RQ-1: Lack Of uniform representation of cross-layered XaaSs	The state of the art analysis has shown that there is a lack of a uniform representation for cross-layered XaaSs that unifies all views on an XaaS, e.g. from the customer view on the APIs and SLA, to the developers that are responsible for deploying and maintaining the XaaS through the cloud.
RQ-2: Lack of considering the appealing characteristics of the cloud as a deployment environment for the SaaSs	Although cloud-based development may benefit from adopting the SOA principles and techniques by following the SOA developments methodologies, none of these methodologies consider the appealing characteristics of the cloud as a deployment environment for the SaaSs.
RQ-3: Lack of a concrete definition language for SaaS applications	Although there are some useful reference architectures for developing cloud-based SaaSs following the SOA paradigm, a lot of supports are lacking here including a concrete definition language for the SaaS applications and components, a mapping approach for finding necessary cloud resources, and the ability to specify and resolve end-to-end constraints of SaaS applications that might affect the underlying cloud resources.
RQ-4: Lack of controlled support and optimisation for end-to-end services	The cross-organisational nature of service systems and the potential composition of services across organisational boundaries requires that services are appropriately designed and effectively managed end-to-end for operational and performance effectiveness. In service eco-systems end-to-end services should be configured or re-configured according to QoS parameters, service preferences and requirements declared either by software developer or contained in the terms of an agreed upon SLA.
RQ-5: Lack of matching service design options with infrastructure	The volatile requirements of service-based applications place demands that the execution infrastructure be appropriately configured in response to application characteristics, end-to-end QoS requirements, or when further functional optimisation is required. Research is required on virtualisation techniques for cross correlating service components at the application-level with the most appropriate platforms and infrastructure.
RQ-6: Lack of achieving scalability in service eco-systems	Smart Internet service eco-systems require a scalable infrastructure that can be scaled up or down based on application demand, levels of QoS and availability of resources, dynamically evolving workloads, while maintaining critical architectural constraints.
RQ-7: Lack of a unified service/cloud service engineering methodology	This item is the common denominator of all previous open research problems. There is a clear necessity for modern service engineering approaches to infuse cloud computing concepts and functionality into service-oriented systems. In this way it is possible to take a unified holistic view of the complete service-system solution lifecycle that causally connects high-level decisions at the application-level down to the level of resource virtualisation and provisioning of physical resources.

tified while evaluating and comparing the different approaches in the sate-of-the-art. Most of the research activities contributing to the state-of-the-art described in the previous section concentrate on platform/infrastructure resource provisioning and attempt to combine and optimize interrelated PaaS and IaaS resources. However, we observe little research work on the cloud application (SaaS) level that supports the development of SBAs by utilizing distributed SaaS components that are deployed on a federation of elastic and heterogeneous PaaS and IaaS resources. Current approaches for cloud-based SBA development cannot meet this expectation and usually leads to a vendor lock-in approach, where the constituting monolithic SaaS components are predominantly tethered to proprietary platforms and infrastructure of a cloud vendor. This approach fails to follow the true spirit of SOA that promotes the reuse of loosely coupled services, thus makes it difficult for SBA developers to migrate the SaaS components from one cloud vendor to another or back to an in-house IT environment.

Moreover, existing approaches hardly address end-to-end non-functional requirements, and are not closed-feedback loop, thus partitioning service systems that involve many providers thereby increasing mean time to resolution of errors. For these methodologies scalability, optimal use of resources and continuous improvement of services are hardly considered. Further, they do not address the nature of the execution environment that automates the end-to-end services and their subsequent operation. None of the current methodologies considers the appealing characteristics of the cloud as a deployment environment. This is where our approach differs by addressing the challenges mentioned in Table 1. This table identifies and summarizes open areas of research priority, called research questions (RQs), with large potential for major breakthrough.

4 CONCLUSIONS

This paper provided a survey on existing support for SBA development on the cloud. As a summary, the survey has shown that the current cloud solutions are mainly fraught with the following shortcomings:

- They introduce a monolithic SaaS/PaaS/IaaS stack architecture where a one-size-fits-all mentality prevails. They do not allow developers to

mix and match functionality and services from multiple application, platform and infrastructure providers and configure it dynamically to address application needs.

- They introduce rigid service orchestration practices tied to a specific resource/infrastructure configuration for the cloud services at the SaaS level.

The above points hamper the (re)-configuration and customization of cloud applications on demand to reflect evolving inter-organizational collaborations. There is clearly a need for a SBA development methodology that allows to mash up services from a variety of cloud providers to create what has been termed a cloud ecosystem. This type of integration allows the tailoring of services to specific business needs using a mixture of SaaS, PaaS and IaaS. In this paper, we have also identified in Table 1 the research questions and challenges as a roadmap for future research towards such a novel SBA development methodology.

REFERENCES

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R. H., Konwinski, A., Lee, G., Patterson, D. A., Rabkin, A., and Zaharia, M. (2009). Above the clouds: A berkeley view of cloud computing. Technical report.

Bernstein, D., Ludvigson, E., Sankar, K., Diamond, S., and Morrow, M. (2009). Blueprint for the intercloud - protocols and formats for cloud computing interoperability. In *Proceedings of the 4th ICIW'09*. IEEE.

Brunelière, H., Cabot, J., and Frédéric, J. (2010). Combining model-driven engineering and cloud computing. In *Proceedings of the 4th edition of Modeling, Design, and Analysis for the Service Cloud*.

Cai, H., Zhang, K., Wang, M., Li, J., Sun, L., and Mao, X. (2009). Customer centric cloud service model and a case study on commerce as a service. In *Proceedings of the IEEE CLOUD'09*.

Chapman, C., Emmerich, W., Márquez, F. G., Clayman, S., and Galis, A. (2010). Software architecture definition for on-demand cloud provisioning. In *Proceedings of the 19th ACM HPDC '10*, pages 61–72, NY, USA. ACM.

Chieu, T., Mohindra, A., Karve, A., and Segal, A. (2010). Solution-based deployment of complex application services on a cloud. In *Proceedings of the IEEE SOLI'10*.

Galán, F., Sampaio, A., Rodero-Merino, L., Loy, I., Gil, V., and Vaquero, L. M. (2009). Service specification in cloud environments based on extensions to open standards. In *Proceedings of the 4th COMSWARE '09*, pages 19:1–19:12, NY, USA. ACM.

Hamdaqa, M., Livogiannis, T., and Tahvildari, L. (2011). A reference model for developing cloud applications. In *In proceedings of CLOSER'11*.

Keahey, K., Tsugawa, M., Matsunaga, A., and Fortes, J. (2009). Sky computing. *IEEE Internet Computing*, 13(5):43–51.

Konstantinou, A. V., Eilam, T., Kalantar, M., Totok, A. A., Arnold, W., and Snible, E. (2009). An architecture for virtual solution composition and deployment in infrastructure clouds. In *Proceedings of the 3rd workshop VTDC '09*, pages 9–18, NY, USA. ACM.

La, H. J. and Kim, S. D. (2009). A systematic process for developing high quality saas cloud services. In *Proceedings of the 1st CloudCom '09*, pages 278–289, Berlin, Heidelberg. Springer-Verlag.

Maximilien, E. M., Ranabahu, A., Engehausen, R., and Anderson, L. C. (2009). Toward cloud-agnostic middlewares. In *Proceeding of the 24th ACM SIGPLAN OOPSLA '09*, pages 619–626, NY, USA. ACM.

Mietzner, R. (2010). *A method and implementation to define and provision variable composite applications, and its usage in cloud computing*. Dissertation, Universität Stuttgart, Germany.

Monteiro, A., Pinto, J., Teixeira, C., and Batista, T. (2011). Cloud interchangeability - redefining expectations. In *Proceedings of CLOSER'11*.

Nguyen, D. K., Lelli, F., Taher, Y., Parkin, M., Papazoglou, M. P., and van den Heuvel, W.-J. (2011). Blueprint template support for engineering cloud-based services. In *Proceedings of ServiceWave'11*, pages 26–37.

Papazoglou, M. P. and van den Heuvel, W.-J. (2006). Service-oriented design and development methodology. *Int. J. Web Eng. Technol.*, 2(4):412–442.

Rochwerger, B. and et al. (2009). The reservoir model and architecture for open federated cloud computing. *IBM Journal of Research and Development*, 53(4).

Thrash, R. (2010). Building a cloud computing specification: fundamental engineering for optimizing cloud computing initialtives. CSC Whitepaper.

Tsai, W.-T., Sun, X., and Balasooriya, J. (2010). Service-oriented cloud computing architecture. In *Proceedings of the 7th ITNG'10*, pages 684–689. Ieee.

Vambenepe, W. (2009). Reality check on cloud portability. SD Times.