Scalable Computing: Practice and Experience
Volume 13, Number 3, pp. 215–231. http://www.scpe.org

ISSN 1895-1767
© 2012 SCPE

ON ENGINEERING CLOUD APPLICATIONS - STATE OF THE ART, SHORTCOMINGS ANALYSIS, AND APPROACH

YEHIA TAHER[*], DINH KHOA NGUYEN, FRANCESCO LELLI, WILLEM-JAN VAN DEN HEUVEL, AND MIKE P. PAPAZOGLOU

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 the 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. 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, methodology, software and product 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 [25].

Key words: Cloud Computing, Service-based Application (SBA), Service-oriented Architecture (SOA), Cloud Development Methodology

1. INTRODUCTION.

Software service technologies aim to support effective combination of independent software services, which are made available by diverse providers, into end-to-end service aggregations on a global scale and in much more powerful and innovative ways that respond to the needs of any kind of service consumers. Central to this aim is the concept of *Service-Oriented-Architecture* or *SOA* [29]. SOA 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.

The enterprise information system of the future will comprise of unbounded numbers and combinations of service eco-systems, which are network-structured, software-intensive, geographically dispersed, and have a global reach. A *service eco-system* is a system of systems [15], which depends on distributed control, cooperation, influence, cascade effects, orchestration, and other emergent behaviours as primary compositional mechanisms to achieve its purpose. The purpose, structure, and number of components in a service eco-system are increasingly unbounded in their development, use, and evolution. Service eco-systems support the development of end-to-end services and SBAs through the creation of alliances between service providers, through which each offered service can be used or syndicated with other services. For instance, an integrated logistics application, which entails the management of an entire logistics chain as a single application in the form of integrated services, could comprise many end-to-end services, such as procurement, order management, production planning and scheduling, inventory control, etc. Typical consumers of such service eco-systems are private organizations, public entities, user-communities, or any combinations of these. Flexible service infrastructures and technologies that can be used as the foundation for developing service eco-systems and applications are gradually providing the incentive and are paving the way for business and social innovation.

However, a serious limitation of an SOA is that it does not make any assumptions regarding service deployment and 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-based application 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.

[*]European Research Institute in Service Science, Tilburg University, 5000 LE, Tilburg, The Netherlands, (Y.Taher@TilburgUniversity.edu).

To address these serious shortcomings it is normal to turn our attention to *Cloud Computing* as it aims to provide both the economies of scale of a shared infrastructure and a flexible delivery model that naturally complements the service orientation of the SOA paradigm. 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 [28]. Cloud capabilities are defined and 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. And this begins to explain why the service orientation provided by an 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 a complete services-based solution. 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 three levels support the virtualisation and management of different levels of the computing solution stack as follows:

- IaaS: is the delivery of hardware (server, storage and network), and associated software (operating systems virtualisation technology, file system), as a service. It is an evolution of traditional hosting that does not require any long-term commitment and allows users to provision resources on-demand. IaaS incorporates the capability to abstract resources as well as deliver physical and logical connectivity to those resources and provides a set of APIs that support the interaction with the infrastructure by consumers. The full power of IaaS can only be used if the flexibility of IaaS deployment and resource allocation has already been considered during the design and development of service-based applications; something that is not possible with today's IaaS approaches. Amazon Web Services Elastic Compute Cloud (EC2) and Secure Storage Service (S3) are well-known examples of current IaaS offerings.
- PaaS: is an application development and deployment platform delivered as a service to developers over the Web. PaaS facilitates development and deployment of applications without the cost and complexity of buying and managing the underlying infrastructure. PaaS offerings comprise of infrastructure software, and typically include a database, middleware and development tools for delivering Web applications and services from the Internet. The consumer's application, however, usually cannot access the infrastructure underneath the platform.
- SaaS: is an "on-demand" application delivery model over the Internet built upon the underlying IaaS and PaaS stacks. It provides a self-contained operating environment used to deliver the entire user experience including the content, its presentation, the application(s), and management capabilities. The SaaS consumer can only access the exposed functions of the application. A typical example is SalesForce.com that offers CRM applications accessible by subscription over the Web. SaaS provides the most integrated functionality built directly into the offering with no option for consumers' extensibility. It cannot handle application variabilities and does not follow the "true" spirit of the SOA paradigm.

The shortcomings and rigidity of current cloud computing service offerings highlighted above prohibit the development of flexible cloud-enabled SBAs. These limitations can be addressed when combining the SOA principles with cloud services resulting in a mixing and matching of external services with on-premise assets and services. This merger offers unprecedented control in allocating resources dynamically to meet the changing needs of SBAs, which is only effective when the service level objectives at the application level guide the cloud's infrastructure management layer. The combination of an SOA with cloud computing concepts brings together scalability, speed, modularity, reuse and the choice of the most appropriate implementation and deployment infrastructure for end-to-end services. Those basic principles are going to allow us to develop novel approaches that revolutionize the way service development is conducted by giving rise to the notion of smart Internet. Moving successfully into smart services poses challenges and opens up possibilities for pioneering research. Not only does it require novel concepts and techniques that will infuse cloud capabilities into an SOA but also requires that application appropriateness be tested and be optimized against both business and technical characteristics. From a business perspective, the application needs to be evaluated in terms of core functionality, service reuse, QoS, compliance requirements, and Service Level Agreement term stipulations. From a technical perspective, the application needs to be evaluated in terms of usage, performance, latency, service-level needs, execution environment and dependencies. Proper application partitioning and fit leads to better performance, scale, ease of change and efficiency that would not otherwise be possible. Our main contribution in this paper is twofold:

- An extensive state-of-the-art analysis on the supports for SBA development on the cloud, which also

leads to the identification of their shortcomings.
- A high-level description of the requirements of a novel uniform cloud service representation model which aims to realize the aforementioned view of a combination of SOA with cloud computing concepts.

The rest of the paper is organized as follows. Section 2 presents an extensive state-of-the-art evaluation. Then, Section 3 identifies the shortcomings of existing approaches and highlights some research challenges derived from the state-of-the-art evaluation. To target the identified shortcomings and research challenges, Section 4 presents a contemporary approach on supporting the SBA development on the cloud. Finally, Section 5 concludes the paper.

2. STATE-OF-THE-ART AND EVALUATION. An SOA promotes loosely-coupled and interoperable service components that are easily shared within and between enterprises via published and discoverable interfaces [29]. It has been suggested by many experts both in the academia and industry that the SOA paradigm promises many advantages over the other architecture paradigms in terms of reusability, business flexibility and agility, and interoperability. However, current SBA development following the SOA paradigm usually leads to a vendor lock-in approach, where the constituting monolithic service components are predominantly tethered to proprietary platforms and infrastructure of a vendor and thus not customizable, extendable and interoperable, cf. the left side of Fig. 2.1. That is because current SOA developments do not usually put focus on the deployment environment of the constituent service components.

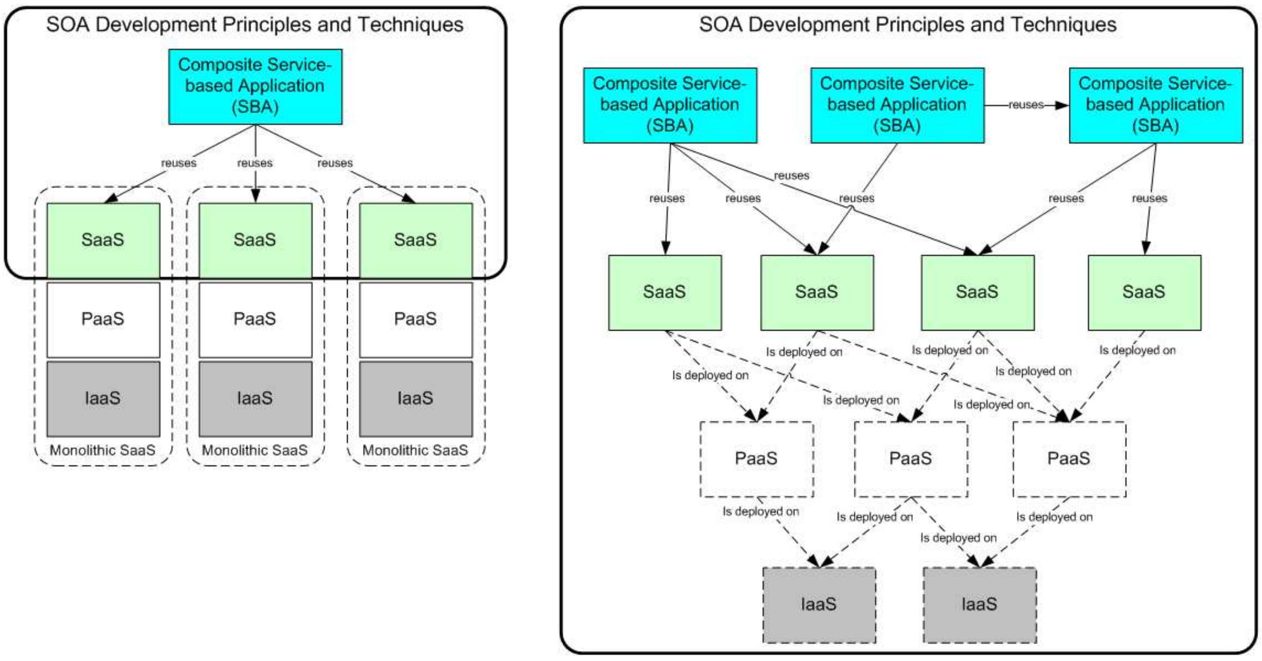
SOA Development Principles and Techniques
Composite Service-based Application (SBA)
reuses
SaaS
PaaS
IaaS
Monolithic SaaS
SOA Development Principles and Techniques
Composite Service-based Application (SBA)
reuses
SaaS
Is deployed on
PaaS
IaaS

Fig. 2.1: Monolithic SBA Development vs. Cloud-based SBA Development following SOA principles and techniques

This limitation can be addressed by breaking the monolithic service offerings into cloud services (XaaSs) across cloud computing layers, i.e. SaaS, PaaS and IaaS, to enable the platform- and infrastructure-agnostic SBA development on the cloud. Following the SOA principles and techniques, SBA developers can reuse and combine distributed cross-layered XaaS functions. The expression and agreement of non-functional properties between XaaS components on the same layer or across layers must also be available so that cloud-based SBAs that have constraints, such as a maximum amount of time or cost, can be created with a guaranteed QoS[1] that is captured in a Service Level Agreement (SLA). These non-functional properties are referred to as XaaS

[1]http://www.s-cube-network.eu/km/terms/q/quality-of-service-qos

Quality Attributes[2] in this document to capture their wide-ranging nature and to illustrate that they can be used to describe any aspect of how XaaS functions are provided across layers.

As can be seen on the right side of Figure 2.1, an SOA-enabling cloud-based SBA development results in an amalgamation of on-premise and external XaaSs that promotes the reusability and composability of XaaSs across SaaS providers and PaaS/IaaS vendors. Our envisioned SBA development methodology should contain the following three features:

- (F1) Platform- and infrastructure-agnostic SBA development on the cloud, i.e. by using the XaaS offerings from multiple SaaS providers and PaaS/IaaS vendors
- (F2) Following the SOA paradigm to promote the reusability and composability
- (F3) Supporting the SLA specification, negotiation and monitoring regarding the XaaS Quality Attributes across layers.

However, cloud computing is a relatively new research area and only a few existing work supports our envisioned cloud-based SBA development methodology. In this section, we review and evaluate the existing approaches regarding the three desired features. Regarding (F1), Section 2.1 discusses the existing efforts on providing a standardized XaaS representation for supporting the platform- and infrastructure-agnostic, vendor-independent SBA development on the cloud. Section 2.2 targets both (F1) and (F2) from the methodological point of view by presenting the related approaches to develop SBA on the cloud and the existing SOA development methodologies that provide a set of fundamental SOA principles and techniques to be adhered to during the cloud-based SBA development. Lastly regarding (F3), Section 2.3 reviews the existing PaaS/IaaS software and products for platform- and infrastructure-agnostic SBA development on the cloud, putting emphasis on their SLA support.

2.1. XaaS Standardization Support for Cloud-Based SBA Development. While developing SBAs 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 [33], since cloud computing promises to allow SBA developers to design and develop elastic and inexpensive applications independent of platforms [3]. 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 [24] 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 [34] 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 their 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 SBA development has been studied in [18] that has IaaS, application and deployment orchestrators but falls short of proposing a solution for the problem at hand.

The Distributed Management Task Force (DMTF) group[3] has published standards such as the Open Virtualization Format (OVF) [12], to provide an open packaging and distribution format for virtual machines, and the Virtualization Management (VMAN) [11] specifications that address the 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 supports the specification of 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 [26] 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,

[2]`http://www.s-cube-network.eu/km/terms/q/quality-attribute`
[3]`http://dmtf.org/`

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.

Document [6] 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 (www.occi-wg.org) has been developing a uniform API specification for remote management of Cloud Computing infrastructure [27]. This will support 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-based SBA development from a different perspective, the Model Driven Engineering (MDE) research community has realized the benefit of combining MDE techniques with application development and suggested combining MDE with cloud computing [7]. 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 applications. Following the MDE vision,the authors in [17] propose 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 support the description of the capabilities, technical interfaces, and configuration data for the virtualized infrastructure resources of the cloud application service. Similarly, the work in [8] 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. in [32], 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 [1]. 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.

2.1.1. Summary and Evaluation. In summary (cf. Table 2.1), existing approaches mostly target the infrastructure levels and cover 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. Methodology Support for Cloud-Based SBA Development. 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 SBAs on the cloud. As mentioned before, cloud-based SBA development may benefit from existing SOA development methodologies that aim to provide guidelines, models, best practices, standards and reference architectures necessary to construct a well-defined and well-structured SOA. Hence, Section 2.2.1 reviews some well-known SOA development methodologies. Since an SBA can be considered as a composite SaaS application, in Section 2.2.2 existing methodologies for SaaS development on the cloud are discussed, some of which were already developed based on the XaaS description standards mentioned in the previous Section 2.1.

2.2.1. SOA Development Methodologies. In general, an SOA methodology is either an industrial initiative or academic proposal. First, several industrial SOA initiatives have emerged over the past years, resulting in many methodologies, notably the Service-Oriented Modelling and Architecture (SOMA) [4], Service Lifecycle Process [31], and Service-Oriented Modelling Framework (SOMF) [5]. In addition, vendors such as SAP and BEA, and firms providing analysis and design consulting services such as Cap Gemini, and Everware-CBDI proposed their own methodologies. In academia, we have witnessed several activities that concentrate around SOA analysis, design and development. Most prominent approaches are the Service Development Lifecycle

Table 2.1: XaaS Standardization Support - Summary and Evaluation

Approaches	**SOA-based**	**SLA-aware**	**Maturity Level**			
			Academic Proposal	**Standard**	**Vendor proprietary**	**Industrial Adoption**
SaaS Application Meta-Model [17]		X	X			
Customer Centric Model for SaaS Development [8]		X	X			
OVF [12], [16], [9]		X		X	X (Originally by VMware)	X
SDD [26]				X (by OASIS)		
InterCloud [6]			X			
OCCI [27]				X (By the Open Grid Forum)		
Amazon Cloud-Formation [1]		X (partly)			X (by Amazon)	

(SLDC) methodology [29], SOA Analysis and Design/Decision Modelling (SOAD) [35] and SOA Framework for Service Definition and Realization (SOAF) [14]. It goes far beyond the scope of this article to review all the SOA methodologies in detail here. Hence, we refer to the work in [2], which provides detailed surveys, comparisons and evaluations of a number of existing approaches.

Although cloud-based SBA development may benefit from adopting the SOA principles and techniques by following the SOA development methodologies, none of these SOA methodologies have considered the appealing characteristics of the cloud as a deployment environment for the SBA as a whole, nor for its constituting SaaS components. However, if we look specifically at the development of each individual SaaS component, some of the contemporary approaches have already taken into account the cloud advantages for SaaS developments. As we consider an SBA as a composition of SaaS components following the SOA paradigm, it is worth to understand how these contemporary approaches have recognized the benefit of combining SOA with cloud computing. The next Section 2.2.2 reviews these approaches.

2.2.2. SaaS Development Methodologies. A systematic process for developing high-quality cloud SaaSs has been proposed by La et al. in [20], 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 not discuss about the cloud support for the deployment environment of the SaaSs.

Maximilien et al. introduces a cloud-agnostic middleware in [21] 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 [33]. Using the SOCCA, developers can build SaaS 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 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.

The Cafe application and component templates in [22] 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. Furthermore, application component and cloud resource discovery in Cafe uses WS-Policy matching from [23] and cannot retrieve components and resources that satisfy end-to-end quality-of-service constraints.

The work in [16] 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. The work in [9] 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 support 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 [30] 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 SaaSs on cloud infrastructure. For instance, the authors in [19] 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, [10] describes a solution-based provisioning mechanism using composite appliances to automate the deployment of complex SaaSs on a cloud infrastructure.

2.2.3. Summary and Evaluation. The state of the art analysis (cf. Table 2.2) 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.

2.3. Cloud-Software & Product Support for the Cloud-Based SaaS Development. This section describes the current state-of-the-art of cloud software and IaaS and PaaS cloud offerings for SaaS development, with emphasis on their SLA specification, negotiation and monitoring supports. The expression and agreement of IaaS and PaaS quality attributes are a necessary prerequisite to allow SaaS applications with a minimum quality of service to be built using IaaS/PaaS components. The quality attributes for a service are gathered together in a contract between the customer and a provider, known as a Service Level Agreement (SLA). The section is split into three parts: first, we describe selected IaaS and PaaS cloud computing software (i.e., frameworks that can be used by cloud service providers to offer an IaaS/PaaS service) and show how most of these do not support the expression of quality attributes and SLAs. The second and third parts describe IaaS and PaaS cloud offerings that support SLAs and what quality attributes are used to guarantee the SLA offered. This section concludes with a summary of the findings. Please note that this section concentrates on some of the main products in this category and their features to illustrate the state-of-the-art and is not intended to be a comprehensive catalogue of all available cloud computing products and platforms.

Table 2.2: Cloud-based SaaS Development Methodology - Summary and Evaluation

Approaches	SOA-based	SLA-aware	Maturity Level			
			Academic Proposal	Standard	Vendor proprietary	Industrial Adoption
SOCCA [33]	X	n/a	X			
Café [22]	X	X	X			
Reservoir [30]		X	X			
Cloud-Agnostic Middleware [21]		X (partly)	X			
Virtual Appliance Model [19]			X		X (by IBM)	
Composite Virtual Appliance [10]			X			

2.3.1. Cloud Software Components & Frameworks. Many software frameworks, tools and components have been developed to support the deployment of clouds by third-party (i.e., cloud service providers), however most of them do not offer support for quality metrics or SLAs as described below. As each framework has different priorities they are difficult to classify, therefore we have used the broad criteria of "SLA-enabled" and "non-SLA enabled" to indicate frameworks supporting the expression, monitoring and/or support for quality attributes and those that require additional components to do so:

- **SLA-enabled Components & Framework**

SLA@SOI[4] is an EC-funded FP7 research program to provide "a business-ready service oriented infrastructure empowering the service economy in a flexible and dependable way". A main requirement of "business-ready" includes "Transparent SLA management Service level agreements (SLAs) for defining the exact conditions under which services are provided/consumed can be transparently managed across the whole business and IT stack". The goal of the architecture is to provide a framework such that integration with other Resource managers will quite trivial to achieve. Therefore, it might be the case that the quality metrics supported will be dependent on the metrics supported by the underlying fabric and associated monitoring system.

RESERVOIR[5], sponsored by the EC in the FP7 program, is a software framework for CTOs and CIOs to build cloud infrastructure. RESERVOIR has the goal to deliver better services for businesses and eGovernment with energy-efficiency and elasticity by increasing or lowering computing based on demand. The project "showcases an innovative open SLA management framework" that treats "SLAs as a specific concern within the overall service lifecycle management at infrastructure level". The RESERVOIR component responsible for SLA definition and management, the Service Manager (SM), has been released by Telefonica as the Claudia platform[6], though at the time of writing, the software is not yet finalized and is offered as v0.1.

Claudia[7] supports the TCloud API Self-Monitoring extensions TCloud Self-Monitoring extensions, a set of entities and operations to perform monitoring actions on cloud resources. The extensions allow the monitoring of: CPU, memory and disk usage, and input and output bandwidth for virtual machines, input and output bandwidth for virtual networks, disk usage for virtual disks and CPU, memory, disk and bandwidth usage for virtual data centers (aggregations of virtual machines).

- **Non-SLA-enabled Components & Frameworks**

[4]http://sla-at-soi.eu/
[5]http://www.reservoir-fp7.eu/
[6]http://claudia.morfeo-project.org/
[7]TCloud API v0.9 http://www.tid.es/files/doc/apis/TCloud_API_Spec_v0.9.pdf

Arjuna's Agility can be used by multiple, federated organizations which have their own services, resources and policies. Initially, these organizations are likely to be semi-independent departments within a single enterprise. Ultimately, Agility^{TM} can provide seamless access to external utility/cloud services and resources[8].

Cloud.com provides an open-source cloud-computing platform for building and managing private and public cloud infrastructure[9].

The **Enomaly Elastic Cloud Computing Platform (ECP)** is software that "is specifically designed to meet the requirements of carriers, xSPs, and hosting providers who want to offer an Infrastructure-on-Demand or IaaS service to their customers"[10].

enStratus^{TM} is a cloud infrastructure management solution for deploying and managing enterprise-class applications in public, private and hybrid clouds[11].

Eucalyptus is an open source cloud-computing framework for academic research that provides computational resources for experiment and research. Eucalyptus is tightly integrated with the Xen Hypervisor [13]. It does not provide "metering with centralized reporting, so it's difficult to fulfill Service Level Agreements [using it]"[12].

Kaavo offers software that can handle the scale and complexity of application lifecycle management in the clouds and manage applications running on physical resources from different IaaS providers[13].

Nimbus is an open source toolkit aimed at the science community that supports the conversion of a cluster into an Infrastructure-as-a-Service (IaaS) cloud[14].

OpenNebula is an open-source cloud computing toolkit for managing heterogeneous distributed data centre infrastructures[15]. It "provides an abstraction layer independent from underlying services for security, virtualization, networking and storage, avoiding vendor lock-in and enabling interoperability"[16].

OpenStack is an IaaS cloud-computing project started by Rackspace Cloud and NASA but now contains 60 other companies including Citrix Systems, Dell, AMD, Intel, Canonical and Cisco[17]. OpenStack Compute is open source software designed to provision and manage large networks of virtual machines, creating a redundant and scalable cloud computing platform[18], whilst OpenStack Object Storage is open source software for creating redundant, scalable object storage using clusters of standardized servers to store petabytes of accessible data[19].

VMWare is a developer of virtualization software used by IaaS providers to host operating system instances[20].

The **Xen Cloud Platform (XCP)** is the fully-open sourced and freely-available follow-on of the XenServer application developed by Citrix who provide software "for powering and managing scalable clouds"[21]. XCP provides "an open source enterprise-ready server virtualization and cloud computing platform, delivering the Xen Hypervisor with support".

2.3.2. IaaS Cloud Offerings with Support for Quality Attributes/SLAs. **Amazon EC2** offers a blanket "*one-size-fits-all*" SLA to its customers. The SLA specifies only that the service should be available 99.95% in a Service Year - 365 days from the date of an SLA claim. Any lower and the customer is eligible for a Service Credit[22]. Guaranteed minimum levels of performance and response times by the platform are not available.

AT&T provides a simple, elastic, secure and cost-effective cloud solution that offers storage and compute services. The compute service has a "service level agreement of 99.9% for availability of the infrastructure"[23]

[8] `http://www.arjuna.com`
[9] `http://www.cloud.com/`
[10] `http://www.enomaly.com/Elastic-Computin.457.0.html`
[11] `http://www.enstratus.com/page/1/about-us-overview.jsp`
[12] `http://cnsa-cloud-project.wikidot.com/eucalyptus\#toc8`
[13] `http://www.kaavo.com/products-and-services/product`
[14] `http://www.nimbusproject.org/`
[15] `http://en.wikipedia.org/wiki/OpenNebula`
[16] `http://blog.opennebula.org/?p=683`
[17] `http://en.wikipedia.org/wiki/OpenStack`
[18] `http://openstack.org/projects/compute/`
[19] `http://openstack.org/projects/storage/`
[20] `http://www.vmware.com/solutions/cloud-computing/index.html`
[21] `http://www.citrix.com/English/ps2/products/product.asp?contentID=1681633&ntref=hp_cat_cloud`
[22] `http://aws.amazon.com/ec2-sla/`
[23] `https://portal.synaptic.att.com/caas`

, whilst the storage service is “backed by a 99.9% service level agreement for availability of the web services API”[24].

GoGrid provides scalable cloud infrastructure in multiple data centres using dedicated and cloud servers, elastic hardware load balancing and cloud storage[25]. The SLA for these services covers the following elements of the service: server uptime, persistent storage, internal and external network performance, load balancing, cloud storage, server reboot, support response time, domain name services, physical security and 24 x 365 engineering support.

IBM Cloud Infrastructure allows its customers to run operating system instances (currently RedHat Enterprise, Suse Linux Enterprise and Microsoft Windows Server) and together with a set of Operations Support Systems (e.g. monitoring and co-ordination software) and optimization services (e.g. support for high performance computing and special processor architectures)[26]. The SmartCloud Enterprise infrastructure cloud “comes with a 99.5 per cent uptime service level agreement”[27].

Joyent provides a cloud platform for creating and deploying scalable applications[28]. The cloud platform contains Smart Machines (compute instances) and specializations called Smart Appliances (e.g., relational and key-value storage and load-balancing). The service level agreement provided by Joyent states that their goal is “to achieve 100% availability for all customers”[29]. However, the SLA provides several caveats for the provider including the ability to scheduled maintenance, emergency maintenance and upgrades at their discretion.

Netmagic Solutions are a managed hosting business based in India that offers the SimpliCloud IaaS to run operating system instances and provide a storage infrastructure[30]. The SLA provided with the product provides a guarantee of service credits if there is less than “99.99% uptime on customer cloud server instance over a calendar month of usage [where the] server instance availability = 100* (total minutes per month - unscheduled downtime minutes) / total minutes per month”[31].

Online Tech is the supplier of a private cloud hosting service packages[32]. The data centres hosting the IaaS are SAS 70 & SSAE 16 certified environments, which is “validation to our clients and potential clients that our security procedures, change management practices, and the quality of our services is satisfactory”[33]. The managed is advertised with “100% Uptime – High Availability”, but it is not clear if this is part of the SLA which comes with the package[34].

ReliaCloud offers a similar product to Amazon’s EC2 with their Cloud Servers, i.e., customers can launch instances of operating system templates to run their applications and are billed according to the rate for the template and the overall time it was active. The SLA for this offering is “a cloud platform with enterprise-grade reliability and security. All our cloud servers are persistent and highly available [and] if one part of our cloud platform fails, your servers will restart somewhere else in our cloud”[35]. A guarantee on the time between detection of a failure and a service restarting is not given.

Windows Server Hyper-V[36] is a private cloud offering from Microsoft. Unclear if the infrastructure offers a standard (or any) SLA - it is possible the SLAs are negotiated individually as part of the private cloud agreement. Note Microsoft’s Azure PaaS does offer an SLA.

2.3.3. PaaS Cloud Offerings with Support for Quality Attributes/SLAs. There are fewer PaaS cloud services available than IaaS because a PaaS service must provide a flexible yet secure programming/development environment in addition to providing a scalable underlying resource infrastructure.

Commensus provides to its customers a hosting and virtualization platform (C-VIP) in the V-Cloud Enterprise Private Cloud solution. The platform is backed by a “99.999% uptime SLA and consistently achieves

[24]https://portal.synaptic.att.com/staas
[25]http://www.gogrid.com/
[26]http://www.ibm.com/cloud-computing/us/en/
[27]http://www.theregister.co.uk/2011/04/08/ibm_smartcloud_enterprise/
[28]http://www.joyentcloud.com/
[29]http://joyent.com/company/policies/cloud-hosting-service-level-agreement
[30]http://www.netmagicsolutions.com/cloud-hosting-services/
[31]http://www.netmagicsolutions.com/service-level-agreements.html
[32]http://www.onlinetech.com/cloud-computing-hosting/private-cloud-hosting-packages
[33]http://www.onlinetech.com/secure-hosting/certified-data-centers/sas-70
[34]http://www.onlinetech.com/cloud-computing-hosting/managed-cloud-hosting
[35]http://www.reliacloud.com/cloudservers/
[36]http://www.microsoft.com/en-us/server-cloud/windows-server/server-virtualization.aspx

100% network uptime"[37].

The **Google App Engine**'s SLA is described as "a draft of proposed SLA terms only. These proposed terms are not applicable to existing Google products and services in any way. Google reserves the right to modify these SLA terms at any time at its discretion" and provides generous exclusions for the provider[38]. However, the agreement does describe SLA metrics for "error rates" (number of Error Requests divided by the total number of user requests to a particular Customer application during a given five minute period), "error requests" (any user request to Customer's application that results in an error, which is caused by the Service, from the application server infrastructure or application datastore infrastructure in response to a valid read or write request) and "Monthly Uptime Percentage" (100% minus the average of the Error Rates from each five minute period in the monthly billing cycle for a particular Customer application).

OrangeScape[39] offers a PaaS that builds on top of Google's App Engine (see above) and provides an environment for building rich applications. Does offer a feature to define application-level SLAs on individual activities (steps in the application workflow accomplished through completing one or more actions), but no SLA for the platform is provided.

Windows Azure has separate SLAs for compute and storage instances, SQL platform and application fabric[40]. The compute SLA offers a monitoring assurance that "99.9% of the time we will detect when a role instance's process is not running and initiate corrective action" and a network availability guarantee "that two or more role instances[41] in different fault and upgrade domains will have external connectivity at least 99.95% of the time". For storage instances, Microsoft "guarantees that at least 99.9% of the time we will successfully process correctly formatted requests that we receive to add, update, read and delete data and that your storage accounts will have connectivity to our Internet gateway". The SQL platform has the guarantee that it will maintain a 'Monthly Availability' of 99.9% during a calendar month. 'Monthly Availability Percentage' for a specific customer database is the ratio of the time the database was available to customer to the total time in a month. Time is measured in 5-minute intervals in a 30-day monthly cycle. Availability is always calculated for a full month. An interval is marked as unavailable if the customer's attempts to connect to a database are rejected by the SQL Azure gateway". The application fabric SLA is the same as that of the compute and storage instances.

WOLF, offered by Wolf Frameworks, is an Online Database Application Platform architected to help the user design, deliver and use Software-as-a-Service (SaaS) database applications[42]. The platform offers "a Service Level Assurance of 99.97% and maintains a strict Business Continuity Service"[43].

2.3.4. Summary and Evaluation. The state of the art in cloud software and product support has shown that several IaaS and PaaS providers support quality metrics and their definition in SLAs. However, in most of these cases, none are negotiable or machine-readable (and mainly concentrate on guaranteeing availability or the uptime of the infrastructure or platform) and there is a lack of support for more advanced quality metrics, for example only Google App Engine includes error rates in the SLA. The survey has also highlighted the lack of support for quality attribute monitoring and management and SLA support in cloud computing components and frameworks (cf. Sect. 2.3.1) that can be used to create new clouds.

3. SHORTCOMINGS OF EXISTING APPROACHES.

This section summarizes the shortcomings we identified while evaluating and comparing the different approaches in the state-of-the-art (cf. Table 3.1).

As a summary, 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 cloud platform (PaaS) and infrastructure (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. Unfortunately, current approaches for cloud-based SBAs development cannot meet this expectation and usually

[37]`http://www.commensus.com/About-Us/Cloud-Platform`
[38]`http://code.google.com/appengine/sla.html`
[39]`http://www.orangescape.com/`
[40]`http://www.microsoft.com/windowsazure/sla/`
[41]A role instance is a.NET program that works with IIS, for example ASP.NET or WCF (Web services).
[42]`http://www.wolfframeworks.com/`
[43]`http://www.wolfframeworks.com/faq.asp`

Table 3.1: Summary of existing gaps and innovations needed to address them

Shortcoming (S)	Research Challenge (RC)	Relevant State-of-the-art Survey
S-1: Lack of uniform representation of cross-layered XaaSs	**RC-1**: 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.	XaaS Standardization (cf. Sect. 2.1)
S-2: Lack of considering the appealing characteristics of the cloud as a deployment environment for the SaaSs	**RC-2**: 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.	SBA development methodology (cf. Sect. 2.2)
S-3: Lack of a concrete definition language for SaaS applications	**RC-3**: 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.	XaaS Standardization (cf. Sect. 2.1)
S-4: Lack of controlled support and optimization for end-to-end services	**RC-4**: The cross-organizational nature of service systems and the potential composition of services across organizational 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.	SLA support of Cloud Software and Product (cf. Sect. 2.3)
S-5: Lack of matching service design options with infrastructure	**RC-5**: 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 virtualization techniques for cross correlating service components at the application-level with the most appropriate platforms and infrastructure.	SBA development methodology (cf. Sect. 2.2) and SLA support of Cloud Software and Product (cf. Sect. 2.3)
S-6: Lack of achieving scalability in service eco-systems	**RC-6**: 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.	SLA support of Cloud Software and Product (cf. Sect. 2.3)
S-7: Lack of a unified service/cloud service engineering methodology	**RC-7**: 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.	SBA development methodology (cf. Sect. 2.2)

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; i.e., existing SaaSs from providers like SalesForce, Google or SaaSDirectory are often not customizable, extendable or interoperable. This approach fails to follow the true spirit of an 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 a 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 vision differs by deriving also the research challenges in the Table 3.1 to address the identified shortcomings. Through the research challenges, the Table 3.1 summarizes the open areas of future research priority with large potential for major breakthroughs.

To give an idea of how to target the research challenges pointed out in the Table 3.1, the next Section 4 presents an ongoing approach for engineering SBAs on the cloud: the blueprinting approach. Following this approach, SBA developers can easily syndicate cross-layered XaaSs from multiple cloud vendors to address their application needs, whilst still adhering to the fundamental SOA design principles.

4. BLUEPRINTING APPROACH. As we already noted, developers at the SBA level need to couple their applications in whole or part with external SaaS offerings to provide opportunities related to other areas of their clients' business. To allow full automation and optimization of SBA level actions requires a causal connection of application-level process operations to configurable supporting platforms and infrastructure services. This approach promotes autonomous services (at all levels of the cloud stack) that adhere to the same principles of separation of concerns to minimize dependencies. It allows any service at any layer to be appropriately combined with a service at the same level of the cloud stack or swapped in or out without having to stop and modify other components elsewhere. At the same time, it should allow multiple (and possibly composed) resource/infrastructure or implementation options for a given service at the application-level. Such considerations lead to a syndicated multi-channel delivery model, where it is contrasted the monolithic cloud stack solutions that permeate the cloud today.

This section introduces an approach to target the aforementioned challenges in building cloud-based SBAs that have been derived in the previous Section 3. By decomposing the usual monolithic SaaS offerings and proposing the concept of *Blueprint* as an abstract, uniform representation of cross-layered XaaSs, the Blueprinting Approach is a novel powerful solution that allows SBA applications to dynamically run on top of multiple alternative cloud platform and infrastructure virtualization solutions. Figure 4.1 illustrates how cross-layered components of an SBA solution can be abstracted and described in a series of SaaS, PaaS, and IaaS blueprints to provide a fast and simplified method for provisioning them as cloud services and composing them to build an SBA. The Blueprinting approach also seeks to simplify the SBA deployment by hiding away the complexity of managing all configurations of the underlying middleware and integrating with optimal PaaS/IaaS options. It also achieves portability across clouds and cloud providers to leverage the benefits of elasticity and scale.

Blueprinting supports a flexible top-down continuous closed-feedback loop service refinement and improvement approach. Application-level decisions regarding virtualized end-to-end services are correlated with and are used to drive resource provisioning and adjust the workload and traffic to automate the dynamic configuration and deployment of application instances onto available cloud resources. From the architecture point of view, the blueprinting approach promotes the two fundamental characteristics of a SOA: the reuse and composition of independent, loosely-coupled XaaSs. It supports the independent layering of components within a typical cloud stack, i.e. the XaaS building blocks of an SBA are now composable and interchangeable. For example, a developer can choose to compose SaaSs from multiple SaaS providers into a coherent and integrated SBA, which the developer can then synthesize with PaaSs from one or more PaaS providers, and deploy on different alternative IaaS clouds.

The blueprint framework interlaces the following inter-related elements:

- A declarative Blueprint Definition Language (BDL), which provides the necessary abstraction constructs to describe the operational, performance and capacity requirements of cross-layered XaaSs.
- A Blueprint Manipulation Language (BPML), which provides a set of operators for manipulating, comparing and achieving mappings between blueprints that are defined in BDL.
- A Blueprint Constraint Language (BCL), which specifies any explicitly stated rule or regulation that prescribes any aspect of cloud service.
- A simple blueprint query language for querying collections of blueprints.

In the following, we will briefly introduce the most important elements of the blueprint framework.

4.1. Blueprint Definition Language. To understand the use of blueprinting consider an external developer (e.g., a virtual service operator or provider) that provides mash-up solutions bundling together different types of interactive telecommunications services (wireless home broadband, multi-channel video programming services, including video-on-demand) and enterprise-grade backend services (rating and broadband billing for cable TV services or fixed broadband services, billing processes for mobile services, invoicing, accounts receivable, and so on). The mashing up of these services creates an assortment of offerings that improve business efficiency and customer appeal. The turnkey service solutions are implemented in an end-to-end fashion and involve multiple service providers.

Service providers publish their offerings, e.g., video-on-demand or interactive TV/open cable applications, using a source blueprint model in a telecommunications marketplace - an efficient online platform that manages the distribution of services and the management of bids, thereby streamlining the procurement process.

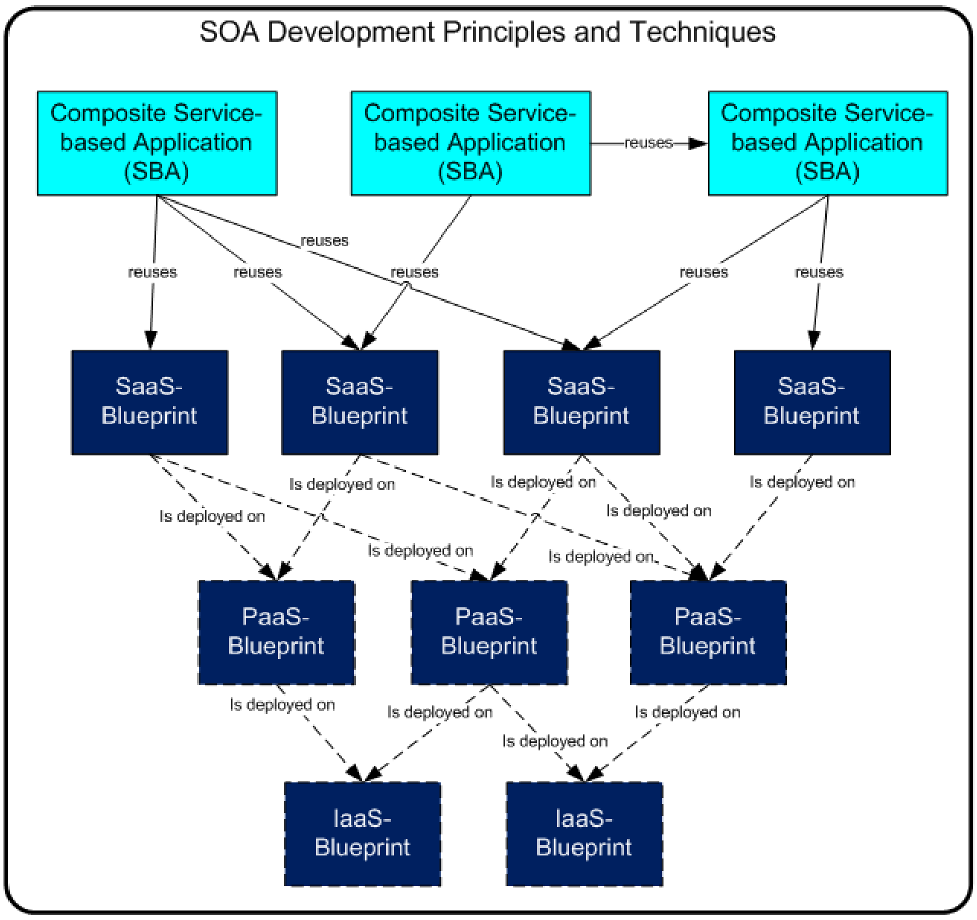

Fig. 4.1: SBA Development using Blueprints

Developers, such as a virtual service provider, can then choose offerings, compose, extend and customize this blueprint model to develop full-featured service-based applications.

A blueprint model is defined in BDL and is based on a clear separation of service processing concerns and is minimally distilled in the following number of inter-related templates:

- Operational service description: This template focuses on the description of functional characteristics of service such as service types, messages, interfaces and operations, namely, the service's signature.
- Performance-oriented service capabilities: This template includes key performance indicators (KPIs) associated with services. Typical examples of quantifiable KPIs are upper and lower performance response time ranges and service availability, throughput, delivery, latency, bandwidth, MTBF (Mean Time Between Failure), MTRS (Mean Time to Restore Service), and so on.
- Resource utilization: This template describes physical infrastructure and resources that are required to run a particular service described in the blueprint model. In general, this template expresses the workload profile including average and peak workload requirements. For instance, a service provider may declare specific technical features that must be taken into account for its service to operate properly, e.g., the server (disk I/O and network) bandwidth required for true on-demand delivery of streaming media, such as video and audio files. This template can express information packaged using the DMFT?s Open Virtualization Format (OVF).
- Policies: This template prescribes, limits, or specifies any aspect of a business agreement that is required by service application developers to use a particular service. It is typically annotated with service level agreements (SLAs) and compliance rules and includes amongst other things security, privacy and

compliance requirements.

In [25], the blueprint XSD template has been released as a first version of the BDL. Within the EC's 4caaSt FP7 project[44] that involves industrial key players in cloud computing such as Telefonica, SAP, Erricson, etc., the blueprint template has been extensively used as a standard, uniform and implementation-agnostic description for XaaS offerings. It has also been validated among the 4caast partners that the blueprint XSD template is capable to capture all the necessary aspects of an industrial cloud service and simple enough to be used by the key industry players in cloud computing.

4.2. Blueprint Constraint Language. The purpose of BCL is to formally express diverse types of policies, such as SLA terms, deployment constraints, data residency constraints, auditability constraints, security constraints, that are related to cloud services and captured by the policy template of the BDL. After a consumer or developer and a provider agree to a set of constraints, these constraints will govern how the services of the provider act.

The BCL is based on a formal foundation to facilitate reasoning and verification by ensuring that services comply with regulations and rules that demarcate their operational behaviour. For this purpose we may use Linear Temporal Logic (LTL) as the formal foundation of BCL and associated compliance patterns. By using automated verification tools that are associated with LTL (e.g. the SPIN model-checker), we can detect compliance violations and provide compliance support for cloud configurations, deployment models and plans that are generated by the blueprint framework prior to execution.

BCL could be used for instance to express that interactive telecommunications services involve high traffic spikes that require automatic provisioning of service instances to accommodate seasonal, e.g., summer period, peaks in demand. The combination of appropriate information from the policy and resource utilization templates will therefore result in claiming resources in advance to accommodate such high traffic spikes.

4.3. Blueprint Manipulation Language. The blueprint model exposes its information in a manner, which facilitates comparison and simple composition of blueprints to express end-to-end offerings from various providers. The BML is based on a set of model-management algebraic operators, such as match, merge, compose, extract, delete and so on. These operators accept source blueprint templates as input and return a new blueprint template as result.

To exemplify the use of BML consider the use of the merging operator on four source blueprints describing high definition IPTV, video on demand, broadband Internet and VoIP services to create an end-to-end triple play service. This operator will yield a target blueprint model that aggregates all four-blueprint models (and their underlying templates) by defining mappings between them. The operator will ascertain that the four blueprint models can be matched to each other and those constraints or capacity requirements are not violated by the target blueprint model.

5. CONCLUSION.

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

- They introduce a monolithic SaaS/PaaS/IaaS stack architecture where a one-size-fits-all mentality prevails. They do not allow SBA developers to mix and match functionalities and services from multiple application, platform and infrastructure providers and configure it dynamically to address their application needs.
- They introduce rigid service orchestration practices tied to a specific resource/infrastructure configuration for the cloud services at the application level.

The above points hamper the (re)-configuration and customization of cloud-based SBAs on demand to reflect evolving inter-organizational collaborations. There is clearly a need to mash up services from a variety of cloud providers to create what has been termed a cloud ecosystem. This type of integration supports the tailoring of SBAs to specific business needs using a mixture of SaaS, PaaS and IaaS.

To deal with the identified shortcomings, we pointed out the need of an abstract and uniform representation for cloud service offerings across cloud computing layers, i.e. SaaS, PaaS, and IaaS. By using this uniform description for cloud service offerings, SBA developers can reuse, customize and combine distributed SaaSs for the SBAs in a seamless manner.

[44]EC's 7th Framework project 4caaSt: http://4caast.morfeo-project.org/

Acknowledgments. 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.

REFERENCES

[1] Amazon Web Services, *Aws cloudformation http://aws.amazon.com/de/cloudformation/*, 2011.
[2] V. Andrikopoulos and et al., *State of the art report on software engineering design knowledge and survey of HCI and contextual knowledge*, Project deliverable PO-JRA-1.1.1, S-Cube Network of Excellence, July 2008.
[3] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andrew Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, and Matei Zaharia, *Above the clouds: A berkeley view of cloud computing*, Tech. report, 2009.
[4] A. Arsanjani, S. Ghosh, A. Allam, T. Abdollah, S. Gariapathy, and K. Holley, *Soma: a method for developing service-oriented solutions*, IBM Syst. J. **47** (2008), no. 3, 377–396.
[5] Michael Bell, *Service-oriented modeling: Service analysis, design, and architecture*, Wiley Publishing, 2008.
[6] D. Bernstein, E. Ludvigson, K. Sankar, S. Diamond, and M. Morrow, *Blueprint for the intercloud - protocols and formats for cloud computing interoperability*, Proceedings of the 4th ICIW'09, IEEE, 2009.
[7] H. Brunelière, J. Cabot, and J. Frédéric, *Combining model-driven engineering and cloud computing*, Proceedings of the 4th edition of Modeling, Design, and Analysis for the Service Cloud, June 2010.
[8] H. Cai, K. Zhang, M. Wang, J. Li, L. Sun, and X. Mao, *Customer centric cloud service model and a case study on commerce as a service*, Proceedings of the IEEE CLOUD'09, 2009.
[9] Clovis Chapman, Wolfgang Emmerich, Fermín Galán Márquez, Stuart Clayman, and Alex Galis, *Software architecture definition for on-demand cloud provisioning*, Proceedings of the 19th ACM HPDC '10 (NY, USA), ACM, 2010, pp. 61–72.
[10] T.C. Chieu, A. Mohindra, A. Karve, and A. Segal, *Solution-based deployment of complex application services on a cloud*, Proceedings of the IEEE SOLI'10, 2010.
[11] DMTF, *Dmtf to develop standards for managing a cloud computing environment, http://www.dmtf.org/standards/cloud.*
[12] *Open Virtualization Format (OVF), http://www.dmtf.org/standards/ovf.*
[13] Patrcia Takako Endo, Glauco Estcio Gonalves, Judith Kelner, and Djamel Sadok, *A survey on open-source cloud computing solutions*, Tech. report, Universidade Federal de Pernambuco, 2010.
[14] Abdelkarim Erradi, Sriram Anand, and Naveen N. Kulkarni, *Soaf: An architectural framework for service definition and realization.*, IEEE SCC'06, 2006, pp. 151–158.
[15] D. A. Fisher, *An emergent perspective on interoperation in systems of systems*, Tech. report, Software Engineering Institute, Carnegie Mellon University, 2006.
[16] Fermín Galán, Americo Sampaio, Luis Rodero-Merino, Irit Loy, Victor Gil, and Luis M. Vaquero, *Service specification in cloud environments based on extensions to open standards*, Proceedings of the 4th COMSWARE '09 (NY, USA), ACM, 2009, pp. 19:1–19:12.
[17] M. Hamdaqa, T. Livogiannis, and L. Tahvildari, *A reference model for developing cloud applications*, In proceedings of CLOSER'11, 2011.
[18] K. Keahey, M. Tsugawa, A. Matsunaga, and J. Fortes, *Sky computing*, IEEE Internet Computing **13** (2009), no. 5, 43–51.
[19] Alexander V. Konstantinou, Tamar Eilam, Michael Kalantar, Alexander A. Totok, William Arnold, and Edward Snible, *An architecture for virtual solution composition and deployment in infrastructure clouds*, Proceedings of the 3rd workshop VTDC '09 (NY, USA), ACM, 2009, pp. 9–18.
[20] Hyun Jung La and Soo Dong Kim, *A systematic process for developing high quality saas cloud services*, Proceedings of the 1st CloudCom '09 (Berlin, Heidelberg), Springer-Verlag, 2009, pp. 278–289.
[21] E. Michael Maximilien, Ajith Ranabahu, Roy Engehausen, and Laura C. Anderson, *Toward cloud-agnostic middlewares*, Proceeding of the 24th ACM SIGPLAN OOPSLA '09 (NY, USA), ACM, 2009, pp. 619–626.
[22] Ralph Mietzner, *A method and implementation to define and provision variable composite applications, and its usage in cloud computing*, Dissertation, Universität Stuttgart, Germany, August 2010, p. 369.
[23] Ralph Mietzner, Tammo van Lessen, Alexander Wiese, Matthias Wieland, Dimka Karastoyanova, and Frank Leymann, *Virtualizing services and resources with probus: The ws-policy-aware service and resource bus*, Proceedings of the ICWS'09, 2009, pp. 617–624.
[24] A. Monteiro, J.S. Pinto, C. Teixeira, and T. Batista, *Cloud interchangeability - redefining expectations*, Proceedings of CLOSER'11, 2011.
[25] Dinh Khoa Nguyen, Francesco Lelli, Mike P. Papazoglou, and Willem-Jan van den Heuvel, *Blueprinting approach in support of cloud computing*, Future Internet **4** (2012), no. 1, 322–346.
[26] OASIS, *Solution deployment descriptor specification 1.0, http://docs.oasis-open.org/sdd/v1.0/os/sdd-spec-v1.0-os.pdf*, Tech. report, OASIS, September 2008.
[27] OCCI-Working Group, *Open cloud computing interface - infrastructure*, April 2011.
[28] Michael P. Papazoglou and Willem-Jan van den Heuvel, *Blueprinting the cloud*, IEEE Internet Computing **15** (2011), no. 6, 74–79.
[29] Mike P. Papazoglou and Willem-Jan van den Heuvel, *Service-oriented design and development methodology*, Int. J. Web Eng. Technol. **2** (2006), no. 4, 412–442.
[30] B. Rochwerger and et al., *The reservoir model and architecture for open federated cloud computing*, IBM Journal of Research and Development **53** (2009), no. 4.
[31] SOA Practitioners Guide Part 3, *Introduction to service lifecyle*, 2006.
[32] R.W. Thrash, *Building a cloud computing specification: fundamental engineering for optimizing cloud computing initialtives*,

CSC Whitepaper, August 2010.
[33] Wei-Tek Tsai, Xin Sun, and Janaka Balasooriya, *Service-oriented cloud computing architecture*, Proceedings of the 7th ITNG'10, Ieee, 2010, pp. 684–689.
[34] William Vambenepe, *Reality check on cloud portability*, SD Times, June 2009.
[35] O. Zimmermann, *An architectural decision modeling framework for service oriented architecture design*, dissertation.de, 2009.
(All links in the reference and footnotes were last visited on 11/10/2012)

Edited by: José Luis Vásquez-Poletti, Dana Petcu and Francesco Lelli
Received: Sep 20, 2012
Accepted: Oct. 15, 2012