INTEROPERABILITY AND PORTABILITY OF CLOUD SERVICE ENABLERS IN A PAAS ENVIRONMENT

David Cunha[1,2], Pedro Neves[1] and Pedro Sousa[2]
[1] *Portugal Telecom Inovação, SA, 3810-106 Aveiro, Portugal*
[2] *Centro ALGORITMI / Department of Informatics, University of Minho, 4710-057 Braga, Portugal*
{*david-g-cunha, pedro-m-neves*}*@ptinovacao.pt, pns@di.uminho.pt*

Keywords: Cloud Computing: Platform-as-a-Service: Portability: Service Delivery Platform: Context-awareness.

Abstract: Nowadays, the competition in the telecommunications market is exciting and new entities with value-added services have emerged over the core network of Telecommunications operators (Telcos). These new participants have taken out the operators' relevance since they are entirely agnostic from infrastructure service connectivity. Therefore Telcos, like Portugal Telecom Inovação (PTIN), need to focus on the provision of services to a user's point of view to not become just a dumb-pipe between the consumers and Cloud service providers.

This paper proposes a definition of a distributed architecture that allows developers to create and expose services through a Service Delivery Platform (SDP). The benefit of such Cloud-enabled SDP architecture is the portability of service enablers between Platform-as-a-Service (PaaS) providers through a standardized API. Service developers may thus select the more suitable PaaS offering in order to build on-top applications, based on the performance required by a service. An example of applications which can take advantage from more versatile Cloud platforms, is the delivery of mobile context-aware services that react to both environment and user conditions selecting the right type of content (e.g. photos, videos, etc.) to deliver.

1 INTRODUCTION

Cloud Computing introduced the concept of computing as a utility bringing an innovative and significant opportunity for operators in terms of new delivery's models (on demand) and new flexible business strategies (B2B) (Armbrust et al., 2009). It allows consumers and enterprises to use services over the hardware and systems software hosted remotely at providers' datacenters. The analogy is, "If you only need milk, would you buy a cow?" (InternalComputer, 2011). Therefore, to obtain the benefit: making video calls, sending emails, etc. the consumers really need to acquire the required software/hardware? With the Cloud Computing concept the answer is no. The everyday needs of the general community can be delivered on a pay-per-use model with only a connection to Internet, just like we do at home with the electricity grid. All the software and hardware is provisioned and governed by the Cloud services providers based on the users' requirements and so allowing a more efficient computing (Zhang et al., 2010). It is thought that one day Cloud Computing will be the 5th utility (after water, electricity, gas, and telephony) (Buyya et al., 2009), and it is currently a popular topic within the academic research community.

A previous paradigm laid the foundations of Cloud Computing with the principles of service orientation. Service-oriented architecture (SOA) facilitates the design, reusability, deployment and management of extendable applications towards interoperable services (Azeez et al., 2010), (K.Ramana et al., 2011). Nowadays, Web Services became the prevailing SOA implementation paradigm over standard Internet protocols (Blum et al., 2009). Applications can be developed as compositions of service enablers distributed through a Service Delivery Platform (SDP) exchanging data with each others. Cloud Computing embraced all these technologies creating a new and promising landscape for a next generation of SOA deployment. Cloud delivery models generally fall in three categories: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS), but only higher Cloud stack levels, such as, SaaS and PaaS will be discussed in this paper. For NIST (U.S. Government's National Institute of Standards and Technology), it can be assumed that with SaaS the consumer uses an applica-

tion but does not control the operating system, hardware or network infrastructure on which it's running. On the other hand with PaaS, the developer uses a hosting environment Service Execution Environment (SEE)/Service Creation Environment (SCE) for controlling his services (life cycle) but does not control the operating system, hardware or network infrastructure on which they are running (NIST, 2011). Underlying these concepts exist financial incentives that attract the both sides: Cloud service providers and consumers, such as enterprises. Service providers charge consumers for the delivery and governance of services, and enterprises reduce both Capital Expenditure (CAPEX) and Operational Expanditure (OPEX) since there is no infrastructure provisioning or software investment to be made at the beginning (Maes, 2010),(Armbrust et al., 2009). Telecommunications operators (Telcos) are interested in taking a share of this Cloud market where only some giants US service providers, namely Google, Microsoft, Amazon and Apple take the lead. To enter the game, operators must exploit their differentiated assets (SDP, OSS/BSS, CRM, service enablers, etc.) enabling a two-sided business strategy with service providers.

With the emergence of numerous providers, the need for interoperability and portability between platforms (PaaS) will enable flexibility for Third-parties' developers. Increasing the Cloud choice will reduce the vendor "lock-in" to specific technologies, such as programming models, software tools or APIs (Parameswaran and Chaddha, 2009),(Dikaiakos et al., 2009). Telcos can explore these potentialities through a Service Delivery Platform that glue the pieces together no matter where they are deployed (Bozman and Chen, 2010). Then, they can sell their service enablers to Third-parties for build-on top applications' in order to play new roles in the Internet value chain. In this context, this work proposes and explains a Cloud-enabled SDP architecture that will simplify the development of services in view to rapidly respond to consumers' needs with the migration of enablers between PaaS providers. Therefore, this paper presents the related work (Section 2), the architecture specification toward a solution (Section 3), a possible usecase in a real mobile environment (Section 4), and the preliminary conclusions (Section 5).

2 RELATED WORK

Recently, some of the Cloud Computing's concerns are associated to interoperability and portability between different providers (Parameswaran and Chaddha, 2009),(Dikaiakos et al., 2009). It should be possible for developers to swap enablers between platforms whenever they need (performance issues, actual costs, etc.) without re-architecting the solutions. This can be a quite challenging with the different data models and proprietary runtime frameworks that each application requires and each provider enables. Currently there are no standards for interoperability or data portability, however some initiatives have emerged from non-profit groups with the collaboration of researchers and some Cloud service providers.

The Open Cloud Manifesto (OCM, 2010) is an initiative supported by various vendors with the vision to standardize Cloud Computing (interoperability, portability, security, governance and management, etc.). The goals of an open cloud, such as, flexibility, speed and agility, etc. are outlined to lead a discussion about new cloud computing paradigms and impacts. Another group, the Cloud Computing Interoperability Forum (CCIF, 2009), proposes to unify Cloud APIs with a standardized semantic interface (Unified Cloud Interface) (UCI, 2009) and layers of abstraction from the underlying infrastructure. The orchestration layer and the federation of Clouds are the features of the CCIF presented architecture. However, Amazon and Microsoft rejected the CCIF agenda. Other initiatives like OGF's Open Cloud Computing Interface (OCCI, 2010) tries to bring, in a quick frame, an API specification for the deployment and monitoring. The current release focus is in the Cloud infrastructures-as-a service (IaaS), however, it is referenced that the model may be suitable for higher interoperability at PaaS and SaaS levels. The DMTF's Cloud Management Working Group (CMWG, 2009) focus on defining architectural semantics to standardize interfaces, interactions and data formats between Cloud environments. This body of work consists in the development of usecases, services life cycle, and an interoperable Cloud architecture for service providers and their consumers and developers.

In this perspective, it takes time for standards to mature, and this lack of acceptance derives from the disregard of providers like Google, Microsoft, Amazon and others, that want to offer differentiated services attracting more customers. On the other hand, defining standards in upper layers like PaaS and SaaS, turns the normalization more dependent of propriety interfaces.

3 TOWARD A SOLUTION

In this section, a Cloud-enabled architecture is proposed clarifying the adopted strategy for the modules integration and inherent development.

3.1 Cloud-enabled SDP architecture

The proposed Cloud architecture is presented in Figure 1 with a private PaaS (PTIN PaaS) and a public PaaS.

At a very basic and abstract level, the SDP can be seen as a collection of service enablers which are orchestrated by a Service Broker and exposed for third-parties' applications development in a SOA paradigm. To consume the available services, the client only needs to use the correspondent Web Services Description Language (WSDL) provided by the Service Broker, which in turn use remote invocation procedure via Simple Object Access Protocol (SOAP) or Representational State Transfer (REST) protocols. Furthermore, the Service Broker is a middleware responsible for tasks such as authentication, authorization, accounting, binding, data transformation, load balancing, monitoring, policies and routing, between deployed services and final consumers. The service enablers may be implemented using any type of technology (Java, Python, PHP, Ruby, C#, Visual Basic, etc.), as long as they are able to communicate with the Service Broker via Web services. Third-parties' developers interact with the Broker through the Developer Interface[1] gaining access to the capabilities offered by the SDP (i.e. the enablers). Then, the permissions required for exposing or managing developed services will be assigned in order to choose the suitable PaaS or deploying an enabler. Afterward, if the developer wishes to change any enabler to another PaaS, the Developer Interface will allow him to do so. The PaaS Manager is the key element from the architecture providing the features required for a more flexible and interoperable PaaS environment. This module will be more detailed in section 3.2.

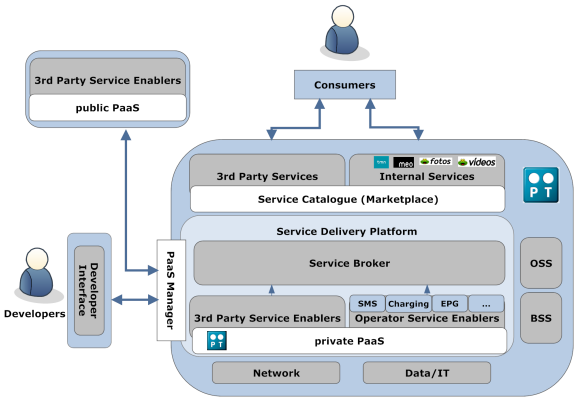

Figure 1: Proposed Cloud-enabled SDP architecture.

[1]wizard platform for service's deployment, migration and management

3.2 Integration Overview

The integration overview presented in Figure 2 focus on the main components of the architecture: the Developer Interface, the PaaS Manager, the Service Broker and the various platforms offering. As mentioned above, the Service Broker has the registration of all available services at the SDP. Querying this component, it is possible to access the list of services and obtain the WSDL that describes each one. For example, when a service is called by a consumer, the Service Broker will invoke the actual service logic which runs in a PaaS environment. The PaaS Manager acts as a standard implementation in order to integrate PaaS offerings with the Developer Interface, allowing the deployment and the portability of service enablers between platforms. Therefore, it is designed with each PaaS APIs implementations which are used for controlling all the offered functionalities. The interoperability between providers can be reached through the PaaS Manager and the Service Broker when a developer manages enablers that are deployed across diverse Clouds.

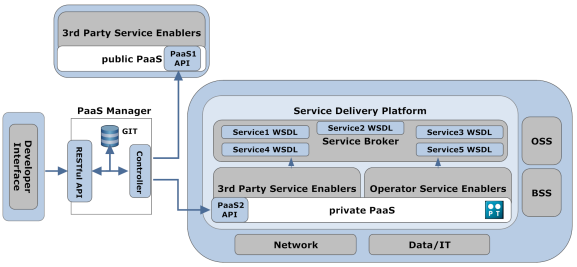

Figure 2: Integration Overview.

3.3 PaaS Manager Developer-Cases

In this section the service registration and service migration processes are now overviewed within the proposed architecture.

3.3.1 PaaS Manager Service Registration

Delving into more technical details, adding a new service to the SDP involves supplying the PaaS Manager with the business logic (i.e. the service code) and the interface of the service (i.e. an Extensible Markup Language (XML) file). The XML interface schema will be handed to the Service Broker, which will register the new service at the SDP. The PaaS Manager component will then deploy the supplied service code at the chosen platform. The final consumer's applications use the WSDL to invoke the service, while the Service Broker invokes the

deployed service code. The defined steps for the register process are the following (see Figure 3):

1 Set the suitable PaaS (PTIN PaaS or public PaaS);

2 Define the interface (XML) and supply the service code (e.g .jar);

3 Register service process at the PaaS Manager;

4 Register service process at the Broker;

5 PaaS Manager deploys the service enabler in PTIN PaaS;

5a Or PaaS Manager deploys the service enabler in a public PaaS.

Figure 3 presents a developer-case of a registration procedure involving the Cloud-enabled SDP architecture components.

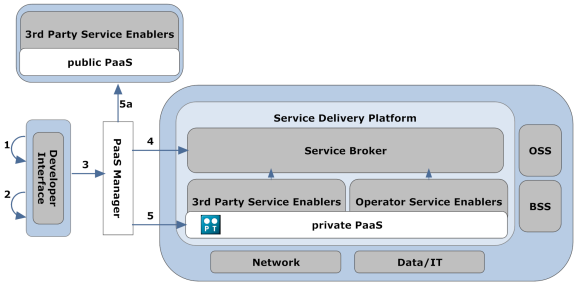

Figure 3: PaaS Manager Service Registration Developer-Case.

3.3.2 PaaS Manager Service Migration

The migration of a service can be quite a challenge. The automated behavior defined requires the un-deployment of the service enabler from the previous platform, the archive being saved in a temporary repository by the PaaS Manager, and finally the deployment in the new PaaS. Meanwhile, the modification is performed at the Service Broker where a new mapping is created in order to locate the new endpoint when the migrated service is called by a consumer application.

The data migration (e.g. databases, etc.) between PaaS is currently a topic under consideration in the proposed architecture. One possible approach is merely migrate databases between providers that offer the same technology, for example, MySQL to MySQL, Oracle to Oracle, etc. Moreover, an additional concern is the dimension size of the services' databases systems, since saving several Terabytes of data in a temporary repository and forwarding it through the network could become a heavy task. The defined steps for the migration process are the following (see Figure 4):

1 Set the new suitable PaaS (PTIN PaaS or public PaaS);

2 Migration service process at the PaaS Manager;

3 Migration service process at the Broker;

4,5,4a,5a Transference service process from previous PaaS, to a temporary GIT repository and finally to the new PaaS;

6, 6a PaaS Manager deploys the service enabler in the new platform;

Figure 4 presents a developer-case of service portability involving the Cloud-enabled SDP architecture components.

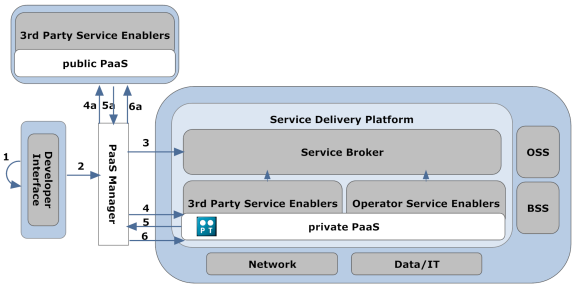

Figure 4: PaaS Manager Service Migration Developer-Case.

4 CLOUD-ENABLED SDP ARCHITECTURE USE-CASE: CONTEXT-AS-A-SERVICE

Context-as-a-Service (CaaS) brings the opportunity to use consumers' generated information for selecting, rate and delivers suitable content (e.g. photos, videos, etc.) to themselves. These metadata can characterize any consumer situation, such as, his position, gender, along with social, music or movies interests. Therefore, context-aware systems proves to be the future of mobile Internet services for finding and grouping users with common interests delivering the most appropriate content (Gomes et al., 2010).

In this section, the proposed use-case shows the ability of automatically selecting the right content to the right people. Public Spaces may easily become Smart Public Spaces by offering a variety of mobile context-aware services that react to both environment and user conditions. The proliferation of such services can, in such cases, generate an overload at some platforms and underlying infrastructures. In order to preserve the adequate performance, the PaaS Manager will allows developers to migrate the service's modules, keeping the process the more agnostic possible for the final consumers. Figure 5 shows the proposed Context-aware architecture integrated with the SDP's Service Broker, a Platform-as-a-Service and a

set of java service enablers. It is important to mention that in the defined architecture some of the CaaS modules are located within the same platform. However, it is the service developer that manages the deployment location of these components.

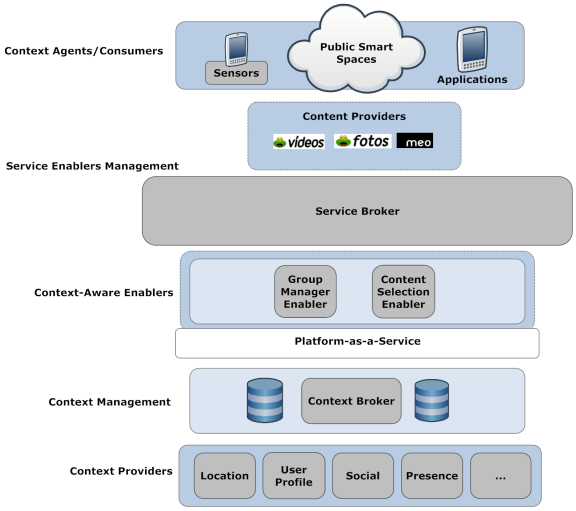

Figure 5: Context-as-a-Service Architecture.

Each CaaS module depicted in Figure 5 has funcionalities that will be detailed briefly:

Context Providers: expose interfaces for providing information to the Context Broker. Through those interfaces the Providers register their availability and capabilities in order to publish different type of retrieved information (e.g. points of interest, calendar, preferences, presence, social, etc.) from a specific user.

Context Broker: responsible for retrieving and delivering information to the Context enablers. It manages and discovers all the registered Context Providers, like the SDP's Service Broker does with the service enablers, matching context elements and storing the information received from the Context Agents.

Group Manager enabler (GME): responsible for identify, create and manage groups of users, delivering context information for further processing. The GME receives the list of subscribed consumers, and starts establishing groups based on configured conditions (e.g. location, social preferences, sensors information, presence, gender, etc.). All this group information is notified to the CSE for the content selection handling process.

Content Selection enabler (CSE): responsible for defining selection rules for the content's ranking deciding which better meets the conditions of a certain user or groups of users. CSE creates a playlist of content and delivers it to the mobile application. This enabler can be more or less complex depending on the multifaceted ranking rules developed.

Content Providers: interact with the consumers' applications delivering the selected content from the playlist (e.g. photos, videos, etc.) by a client.

Context Agents/Consumers: typically located on the consumers portable devices updating client' situation data from several sensors (e.g. location, preferences, presence, social, etc.) for further processing at the Context Broker.

The context consumption is done through XMPP (Extensible Messaging and Presence Protocol) on a publish/subscribe model. XMPP runs on top of TCP/IP protocol stack and it is considered a more interoperable protocol compared to JMS (Java Messaging Service), and a lightweight compared to SOAP (XMPP, 2007),(Gomes et al., 2010). Figure 6 presents the communication scheme.

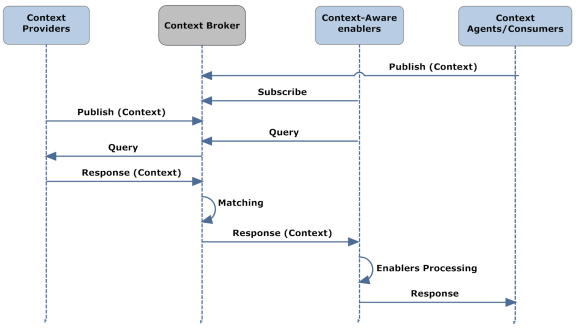

Figure 6: Context Consumption.

5 PRELIMINARY CONCLUSIONS

Cloud interoperability and standardization efforts will increase the quality and diversity of Cloud Computing solutions provided to final consumers. The ability of seamlessly migrate services between providers reduces the existent "lock-in" towards the development of flexible business applications across platforms. This paper presented an architecture that will simplify the development and exposure of services in view to rapidly respond to consumers' needs with the migration of enablers between PaaS providers. However, the databases migration is still under survey since it imposes some dependencies at the PaaS Manager and underlying network. For the architecture prototyping, it was exposed a specification/development approach with the technical requirements and solutions. Currently a PT' Service Delivery Platform solution (Sapo-SDB, 2011) is under exploration for the integration of services in a SOA paradigm.

Moreover, context-aware services are being tested in some platforms providers, namely, Red Hat's Open Shift (Open-Shift, 2011), Google App Engine (GAE, 2011), CloudBees (Cloud-Bees, 2011) and probably a PTIN platform which is under development. The java implementation, and detailed specification of the PaaS Manager, will be soon in progress to integrate and accommodate firstly the PT' SDP, and in a posterior phase, the Developer Interface.

Acknowledgments

The work of P. Sousa was funded by FEDER, through the program COMPETE and the Portuguese Foundation for Science and Technology (FCT), within the project FCOMP-01-0124-FEDER-022674. A partial prototype, namely the context-aware services and the integration with the PT' SDP, will be demonstrated under the scope of the European ICT project "Cloud4SOA" (Contract-No. ICT-257953): (Cloud4SOA, 2010).

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, University of California, Berkeley.

Azeez, A., Perera, S., Gamage, D., Linton, R., Siriwardana, P., Leelaratne, D., Weerawarana, S., and Fremantle, P. (2010). Multi-tenant soa middleware for cloud computing. In *CLOUD '10 IEEE 3rd International Conference on Cloud Computing*, pages 458 – 465, Miami, Florida, USA.

Blum, N., Magedanz, T., Schreiner, F., and Wahle, S. (2009). A research infrastructure for soa-based service delivery frameworks. In *5th International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities*, pages 1–6, Washington DC, USA.

Bozman, J. and Chen, G. (2010). Cloud computing: The need for portability and interoperability. Technical report, Red Hat.

Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., and Brandic, I. (2009). Cloud computing and emerging it platforms vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, 25(6):599–616.

CCIF (2009). The cloud computing interoperability forum. Retrieved December 2011, from: http://www.cloudforum.org/.

Cloud-Bees (2011). Cloud bees. Retrieved January 2012, from: http://www.cloudbees.com/.

Cloud4SOA (2010). The cloud4soa initiative (fp7). Retrieved December 2011, from: http://www.cloud4soa.eu/.

CMWG (2009). Cloud management working group. Retrieved December 2011, from: http://www.dmtf.org/standards/cloud/.

Dikaiakos, M. D., Katsaros, D., Mehra, P., Pallis, G., and Vakali, A. (2009). Cloud computing: Distributed internet computing for it and scientific research. *IEEE Internet Computing*, 13(5):10–13.

GAE (2011). Google app engine. Retrieved January 2012, from: http://code.google.com/appengine/.

Gomes, D., Goncalves, J., Santos, R. O., and Aguiar, R. L. (2010). Xmpp based context management architecture. In *2010 Workshop on Enabling the Future Service-Oriented Internet (EFSOI)*, pages 1372 – 1377, Miami, Florida, USA.

InternalComputer (2011). Understanding what cloud computing means and its advantages. Retrieved December 2011, from: http://www.internalcomputer.com/understanding-what-cloud-computing-means-and-its-advantages.computer.

K.Ramana, Krishna, T., Narayana, C., and Kumar, M. P. (2011). Comparative analysis on cloud computing and service oriented architecture. *International Journal of Advanced Research In Technology*, 1(1):22–28.

Maes, S. H. (2010). Understanding the relationship between sdp and the cloud. In *The First International Conference on Cloud Computing, GRIDs, and Virtualization*, pages 159–163, Lisbon, Portugal.

NIST (2011). The nist definition of cloud computing. Retrieved November 2011, from: http://www.csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf.

OCCI (2010). Open cloud computing interface. Retrieved December 2011, from: http://occi-wg.org/.

OCM (2010). The open cloud manifesto. Retrieved December 2011, from: http://www.openCloudmanifesto.org/index.htm/.

Open-Shift (2011). Red hat open shift. Retrieved January 2012, from: https://openshift.redhat.com/app/.

Parameswaran, A. V. and Chaddha, A. (2009). Cloud interoperability and standardization. *SETLabs Briefings*, 7(7):19–26.

Sapo-SDB (2011). Sapo service delivery broker. Retrieved December 2011, from: http://sdb.sapo.pt/.

UCI (2009). Unified cloud interface. Retrieved December 2011, from: http://www.groups.google.com/group/unifiedcloud/.

XMPP (2007). Xmpp extensions. Retrieved December 2011, from: http://xmpp.org/.

Zhang, Q., Cheng, L., and Boutaba, R. (2010). Cloud computing: State-of-the-art and research challenges. *Internet Services and Applications*, 1(1):7–18.