SCALABLE COMPUTING: PRACTICE AND EXPERIENCE
Volume 13, Number 3, pp. 251–267. http://www.scpe.org

ISSN 1895-1767
© 2012 SCPE

DATASTORES SUPPORTING SERVICES LIFECYCLE IN THE FRAMEWORK OF CLOUD GOVERNANCE

ADRIAN COPIE[†][*] TEODOR-FLORIN FORTIŞ[*][†] VICTOR ION MUNTEANU[*][†], AND VIOREL NEGRU[*][†]

Abstract. While adopting the Cloud, the small and medium sized enterprises (SMEs) could increase their benefit by embracing some Platform-as-a-Service (PaaS) solutions, doubled by a Cloud Governance or Cloud brokerage approach.

However, in order to fully exploit the advantages brought by Cloud environments, and enter in real competition with "big players" from different markets, SMEs must group themselves under the umbrella of a common marketplace, via some Cloud Governance solutions, and expose together complex, tailored and integrated solutions.

The implementation of an effective Cloud Governance solution requires a strong support for storing and manipulating data which is relevant for various aspects of applications, both at business and technical level, support which is closely linked to cloud service lifecycle.

This paper focuses on analyzing the main requirements related to data storage in an effective Cloud Governance system, details the functionality of the most important datastores and presents various use cases involving the entire storage architecture.

Key words: Cloud computing, Cloud governance, Cloud databases

AMS subject classifications. 68M14, 68P15, 68P20

1. Introduction. Built around the five core characteristics, as identified in [21, 25], and over a set of technical characteristics [18], including *service orientation*, *loosely coupling*, *strong fault tolerance*, *link with business models*, *virtualization*, or *ease of use*, Cloud Computing could offer the ideal environment for developing of new models of applications.

In order to fully exploit the new business models offered by the Cloud Computing [33, 34] – by which the Small and Medium-sized Enterprises (SMEs) could ease their migration to the cloud, benefiting from the delegated infrastructure management, high reliability and "pay-as-you-go" economic model – a strong support is required for easy development and deployment of applications (by adopting appropriate Platform-as-a-Service solutions), and (automatically) solve different aspects related with resource management (supported by a cloud management or cloud brokerage solution).

As the *vendor lock-in* problem occurred as a consequence of the large pool of vendor proprietary technologies, and insufficient standardization, different Platform as a Service (PaaS) solutions were considered in order to override the aforementioned difficulties. At the same time, PaaS solutions could enable uniform development, deployment and execution of cloud-enabled applications in various cloud environments.

They are usually built over the Infrastructure-as-a-Service (IaaS) layer, seldom on top of the Hardware-as-a-Service (HaaS) layer, and aims to be the support for the Software-as-a-Service (SaaS) layer, by providing a series of services, as identified in Table 1.1.

One can notice that the majority of the existent PaaS solutions address the resource management and scaling issues (mOSAIC [22, 11, 23, 24], Cloud Foundry [1], OpenShift [2], Morfeo 4CaaSt [3], ActiveState Stackato [4]), offer a development framework agnostic from the cloud providers (mOSAIC, Cloud Foundry, OpenShift, ActiveState Stackate, Morfeo 4CaaSt), address applications lifecycle management (mOSAIC, OpenShift, ActiveState Stackato) or provide a business ecosystem (Morfeo 4CaaSt).

While the different PaaS solutions offer support for applications development and deploying, there are different issues that require special attention through specialized developments, like resource management,service integration, or service orchestration.

The importance of these specialized developments was emphasized in different white papers [12, 8], offering the basic principles for cloud management. Based on these developments, different organizations proposed various *cloud reference models* [20, 30, 27]. However, in order to address the integration and orchestration

[*]West University of Timişoara, Faculty of Mathematics and Informatics, Computer Science Department, Blvd. V.Pârvan nr. 4, Timişoara, Romania (adrian.copie@info.uvt.ro, fortis@info.uvt.ro, vmunteanu@info.uvt.ro, vnegru@info.uvt.ro).

[†]Institute e-Austria, Timişoara (IeAT), Romania

[1]http://cloudfoundry.org

[2]http://openshift.redhat.com/app

[3]http://4casst.morfeo-project.org

[4]http://www.activestate.com/stackato

Service offered	mOSAIC	Cloud Foundry	OpenShift	Morfeo 4CaaSt	Active State's Stackato	WSO2 Stratos
Cloud resource management and scaling	yes	yes	yes	yes	yes	
Framework support		yes	yes		yes	yes
Application services		yes	yes		yes	
Application lifecycle management				yes		
Business ecosystem				yes		

Table 1.1: Services offered by different PaaS solutions

issues, different papers like [12] or [19] recommend a complementary set of services and a mechanism that is able to complement the cloud management limitations called in its entirety *Cloud Governance*.

The present paper is a direct extension of the work performed in [9]. The structure of this paper is as follows: Section 1 offers an introduction in the problematic of the paper. A motivation related to the present work is provided in section 2. In section 3 the architecture of the proposed Cloud Governance framework is detailed, while issues related to the Identity Management in cloud are discussed in section 4. The Services Datastore architecture is discussed in section 5 while the conculsions of the paper are established in section 6.

2. Motivation and related work. Cloud Governance also comes as a solution to the increasing demand of integration of different SaaS offerings [3, 7]. Integration and manipulation of data from various sources is thus an important issue that comes together with Cloud Governance, as a large variety of activities, like service management (publishing, brokering, monitoring, billing), security and others, are required. With a diversity of data types and usage patterns, a unitary approach for data storage cannot offer a feasible solution. With different datastores and catalogues, like the service, template, instance, or policy catalog, as identified in [12, 13], there is a need to identify specific requirements for each of these datastore categories.

This paper focuses on the storage layer in the context of a multi-agent architecture for a Cloud Governance environment aiming to support the service lifecycle, based on the work described in [12], extending previous work performed in [16, 26] for which it details the main components and presents various significant use-cases related to the service lifecycle.

While there is some resembling with the Service Oriented Architecture (SOA) in what concern the services storage paradigm, in service repositories, the present approach treats also the dynamic scaling issues, the services being dynamically distributed over many virtual machines in the cloud. The analyzed architecture pays attention to the issues raised by the privacy management and security as a central pillar in order to offer the SMEs a trustworthy environment in which they can migrate their businesses, in conformity with governmental laws and complying with the business standards.

At the same time, the existent literature and knowledge base related to the current Cloud Governance solutions does not present relevant information related to their persistence layer, so the storage architecture presented in this paper is our original contribution.

2.1. Cloud Management and Cloud Governance. Having a complex environment that allows the orchestration of SME's products in order to offer sophisticated solutions, raises important management challenges. The actual economic context, but also the financial principles lead the cloud providers to look for solutions in order to obtain significant cost savings through standardization of their basic operations. Provisioning the capacity of the servers as they will have an adequate functionality in the same time with an optimal performance is a difficult task which will influence at the end the economic benefits of the cloud providers. These requirements are achieved through Cloud management solutions. According to [15] there are three core features that a cloud management system must fulfill:

- dynamically provisioning or deprovisioning of infrastructure resources
- ensures that the provisioned resources are securely used by the consumers
- offers a centralized way to plan, monitor, report and bill the infrastructure resources in cloud

Cloud Management solutions exist on an emergent market and in the last years many companies tried to provide dedicated products in order to solve specific issues. For example CA Technologies targets its solution called 3Tera [5] to a single platform, while Cloud.com [5] or Enomaly [6] have multiple target solutions for cloud management.

These solutions, however, are not enough when it comes to the enterprise solutions where different workloads must be processed, facing with issues like various ownerships, compliance, security and so on, requiring a Cloud Governance implementation. This implementation must take into account, according to [14] aspects like: *advanced user authentication*, *tight access control*, *encryption* and *budget management*. EnStratus [7] provides a cloud governance solution focused on important aspects of the enterprise cloud like access control, financial control, key management and encryption, logging and auditing. Also, WSO2 developed a framework able to support SOA Governance, called Governance Registry [35].

2.2. mOSAIC project. The FP7-ICT mOSAIC project (Open Source API and platform for multiple Clouds) aims to offer a friendly environment for writing and hosting applications in cloud, by freeing the developers by the 'vendor lock-in' paradigm. The API abstracts all the cloud providers intricacies, providing a unitary development platform and generic infrastructure resources in terms of computation, storage and network. The users of mOSAIC are able to specify their service requirements in a high level language using cloud ontologies and their resources are negotiated and brokered by a multi-agent system organized in a component called Cloud Agency (CA) [31], [32]. Cloud Agency has also responsibilities in terms of cloud management, being in charge with resource provisioning (buying them from cloud vendors), deploying them in the cloud and monitoring at the infrastructure level. CA is able to work together with the mOSAIC's platform or independent.

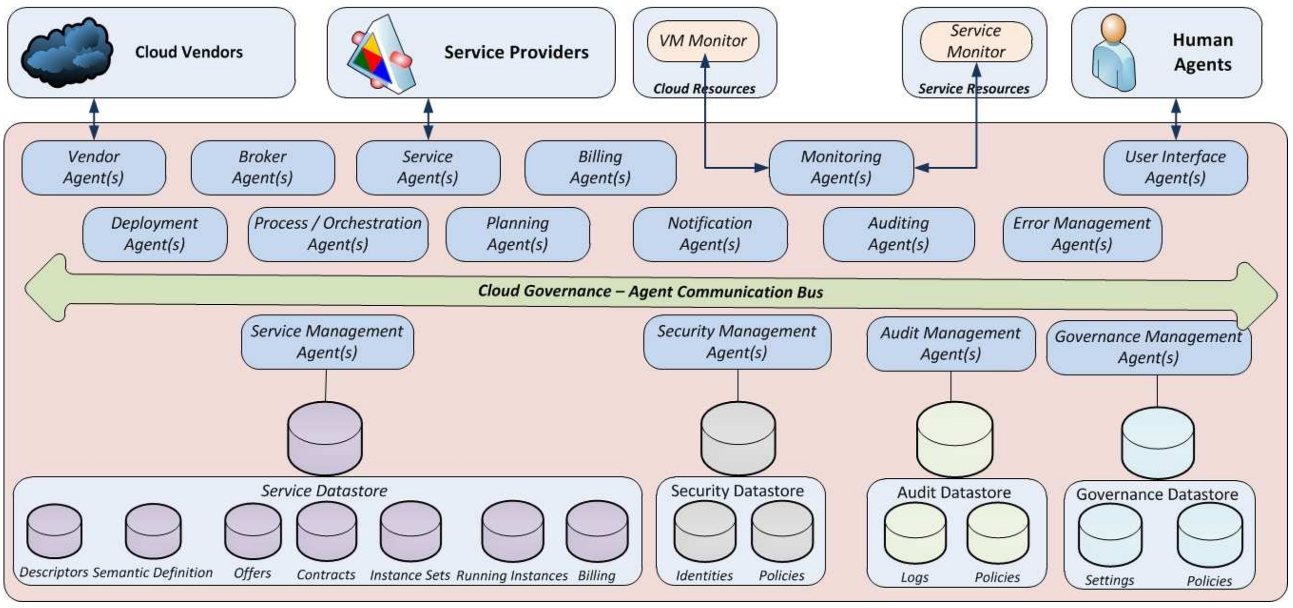

Fig. 3.1: Cloud Governance

[5]http://www.cloudstack.org/
[6]http://www.enomaly.com/
[7]http://www.enstratus.com

3. Architecture of a Cloud Governance system. Using mOSAIC's Cloud Agency component and taking into account the recommendations existing in the Distributed Management Task Force's (DMTF) white papers [12, 13] it is possible to build a Cloud Governance solution able to address the management of services and security, the service monitoring and auditing and offers support for services lifecycle. The proposed solution is a multi-agent system Figure 3.1 that have four specialized interdependent components:

Service Management, that handles all the phases related to the service lifecycle: publish, update, query, remove and instance handling. The requests for publishing a service inside the Cloud Governance Authority environment is directed to the Service Management agency which initiates the appropriate procedures. It is in charge with the service discovery process and takes care of the contracting and instantiation of the services.

Security Management, which deals with all the security issues like identity tracking, generating security tokens and validate them across the system. The Cloud Governance Authority is designed with security in mind, as being a crucial pillar targeted to cloud adoption by the SMEs. This is why a distinct agency dealing with security issues is provided.

The access to resources is performed in a secure way, based on various policies and access control lists, through specialized tokens which are attached to every request. Security Management agency issues the authentication requests to the underneath persistence layer and attaches the generated security tokens to the resource access requests.

Audit Management, for storing and analyzing the audit notifications sent by all the services inside the Cloud Governance environment, being able to raise alerts in case of system malfunctioning

The audit and monitoring inside the Cloud Governance Authority is performed at two distinct levels. The infrastructure level is monitored by the cloud management component which in this case is mOSAIC's Cloud Agency, while the services level is monitored by a specialized agency called Audit Management.

In order to fulfil the SLAs established in the contracting phase, the services are continuously monitored and alarms are raised through Cloud Agency Bus. Audition is also important since, based on the data produced by the audited services, further statistic or functional analyses can be performed.

Governance or *Configuration Management*, which is responsible with the whole system configuration, maintaining information about all the virtual machines running the agents and about the registered services. The Cloud Governance Authority is a complex mechanism composed from many physical or virtual machines and from many software components that are installed and configured to collaborate each other. The information related to the whole ensemble is managed by this dedicated agency which is permanently tied with the cloud management component.

3.1. Cloud Service Lifecycle. In order to have an effective Cloud Governance solution, is essential to offer a service lifecycle functionality which is able to support a service during all its phases in its life. Even that it follows some of the recommendations in [12] and is conceived in terms of Service Oriented Architecture (SOA) the approach of the Cloud Governance system is still different in the approach related to services distribution across many clouds and benefiting of the scaling offered by the cloud environment. When comes to design the architecture for services lifecycle, some important operations must be supported:

- *Design-time* operations: cover the syntactic and semantic service description, allowing the service to be published.
- *Provisioning* operations: cover the offer describing and service publishing, discovery and contracting
- *Deployment* operations: cover the service instantiation and commissioning
- *Execution* operations: cover the services management and billing
- *Retirement* operations: cover the services retirement There is a strong connection between the service lifecycle stages and the Cloud Governance datastores which is revealed in figure 3.2.

Different approaches for service lifecycle management were considered in order to "*manage the dynamic nature of the cloud environment*"[8], or to "*automate the service deployment and the dynamic provisioning of services*"[9] by solutions like BMC Cloud Lifecycle Management, WSO2 Governance Registry, Enstratus DevOps, or Morfeo Claudia. Also, service lifecycle in distributed environments is covered in different papers. In [28] an SCA design model was considered, while in [29] an application-centric lifecycle is offered.

[8]http://www.bmc.com/products/cloud-management

[9]http://claudia.morfeo-project.org/wiki/index.php/Service_Lifecycle_Manager

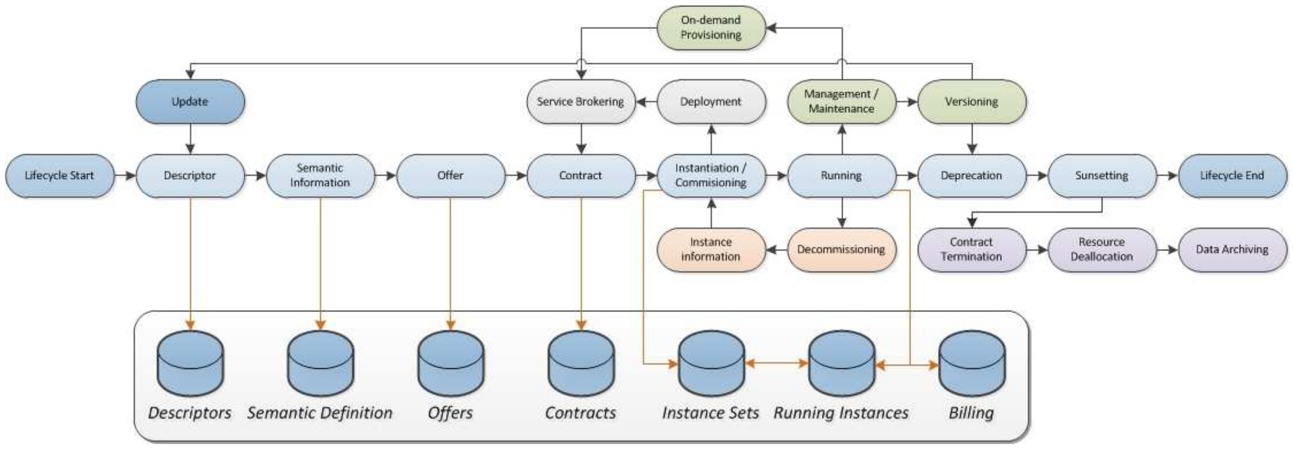

Fig. 3.2: Services Lifecycle

From the Cloud Governance architecture described in figure 3.1 and from the service lifecycle exposed in figure 3.2 four datastores were identified: *Services Datastore*, *Security Datastore*, *Audit and Billing Datastore* and *Governance* or *Configuration Datastore*.

3.2. Databases in Cloud Computing. The large number of services coexisting in a real environment, together with their dependencies, could lead to scalability problems for traditional databases which usually resides on a single server. A relative new scalability technique for relational databases, called sharding, could be used to keep a decent level of performance, but this is not satisfactory because adding or removing one or more server nodes lead to database restructuring, which is time consuming and degrades performance during the operation. In the same time adding more server nodes have as an immediate result the database partitioning, which makes the database system a distributed system.

According to [4], [17] in a distributed system, three essential properties for data cannot be achieved in the same time: *consistency*, *availability* and *partition tolerance*. However, not all the applications need support for all these properties in the same time and based on this assumption, many revolutionary databases have been developed by sacrificing one of the properties for the other two. More than this, some of these new databases, called in opposition to the traditional ACID (Atomicity, Consistency, Isolation, Durability) databases BASE (Basically Available, Soft state, Eventually consistent) are offered in cloud as a service, while some of them are available to be installed as open-source implementations in the private cloud. In this paper, some of the most important database types are analyzed in order to find the best matches for the problems raised by the datastores supporting cloud governance.

3.2.1. Key-Value stores. The key-value stores originate from a paper [10] published by Amazon which proposed a new type of database, highly available and highly reliable that can run on top of commodity hardware and with extremely good scalability while maintaining a high level of performance. Conform to [4] the consistency property is sacrificed for availability and partition tolerance, but this is acceptable for the applications using this kind of database.

The data model is very simple, it consist in a map or a dictionary in which values are assigned or retrieved in correspondence with unique keys, usually represented as strings. The value is perceived by the key-value store as a Binary Large OBject (BLOB) and it is the client's responsibility to interpret the content. The API interface for this datastore is very simple, exposing only few operations: *put*, *get* and *delete*.

The information related to cloud services is very heterogeneous, from structured data describing the services, formatted data resulted from monitoring and auditing processes, to unstructured data and binaries containing the executable code. Storing it in a key-value store involves complex keys management and also requires implementing a transactional mechanism that acts in an all-or-nothing manner for write operations.

Searching a key-value store is not supported by the database itself and it can be performed in a two-step process on the client side: first, all the keys have to be fetched and then their content must be interpreted which is degrades considerably the performance, especially for large data.

3.2.2. Document databases. The document databases represent a key-value stores evolution in terms of the complexity of the information they can store, multiple key-value pairs are allowed to be persisted in a document. The data types supported are the same as for the key-value stores, but the document allows a better information management.

Some of the existing document databases are even ACID compliant [2] fact that assures the data consistency during the Create/Read/Update/Delete (CRUD) operations over the services components. In the same time these database types allow to group more than one service related information inside a document or split it over many documents, offering a primitive linking mechanism between document fields.

3.2.3. Graph databases. The data in a graph database is represented by nodes, links and properties. This representation maps very well to the object oriented applications. The nodes are the object correspondences, where the links signify the relations between objects. Properties are additional information about the objects. The horizontal scalability is extremely good, being able to accommodate with billions of records. The graph databases are suitable for dynamical data since and they not enforce any rigid schema. Many of them are also ACID compliant.

3.2.4. Relational databases in the cloud. The continuous technological advances in cloud computing have shifted the focus to the non relational databases, many 'flavors' being developed, in order to solve specific problems. In the same time, the relational databases have passed through a rethinking process, emphasizing the scalability problems and the opportunity to offer them as a service in the cloud, paradigm called *NewSQL*.

The scalability provided by this paradigm is focused on read operations and is implemented in an elastic manner. From the services inside the cloud governance system, this database imposes constraints related to the structure of the data, which cannot be fulfilled all the time, but on the other side, the searching capabilities are very good supported.

4. Identity management. Cloud computing is an amalgam of technologies which makes possible on-demand resource provisioning with an unprecedented degree of availability and reliability while at the same time allow dynamic scalability. These internal technologies are stacked in various software layers that communicate and provide different services for the adjacent stack level. These layers are heterogeneous, usually they are produced by different vendors. In order to be orchestrated together and offer safe functionality, they have to obey various security rules and protocols implying the use of multiple ways of handling credentials.

This is not a trivial task since the services' environment is the cloud which can scale the infrastructure on multiple servers depending the workload facing the service. These workloads usually have different ownership, needs and compliances which make even harder the management of the entire system. It is the responsibility of a cloud governance system to solve these problems and to provide a unitary framework for the service creators which offer general solutions and for the service consumers which benefit from them.

The services inside a cloud governance system have multiple creators and use various security patterns for user authentication and authorization. More than this, in the most of the times, a customer has multiple accounts and implicitly multiple identities over a multitude of cloud services (e.g. e-bay, Amazon, Gmail, etc). It is an extremely difficult task from the consumer side to use multiple credentials for the different services inside the cloud. Some of them use credentials based on user name and password, some use various secret keys and some are using credit card numbers. A mechanism that can manage all the credentials in a unitary way along all the cloud computing environment is called *identity management* and it is a must for the governance framework.

Identity management is an integral part of cloud governance. Figure 4.1 introduces the identity management architecture used in our proposed cloud governance solution. The presented architecture is shown as high-level components:

The *Client Framework* is a component on the Client side. It facilitates authentication and authorization through a range of methods, tokens and protocols that are used when connecting to one of the *Communication Agents*. It enables communication using standard protocols using various authentication token technologies.

The *Communication Agents* are in charge of providing external communication with the Cloud Governance solution. This component is made out of three agents:

- *Vendor Agent* – is in charge of communicating with all cloud providers. Its main role is provisioning of cloud resources;
- *Service Agent* – is in charge of communicating with service providers as well as service instances. Its main role is to act as a proxy between them and the available cloud governance services;

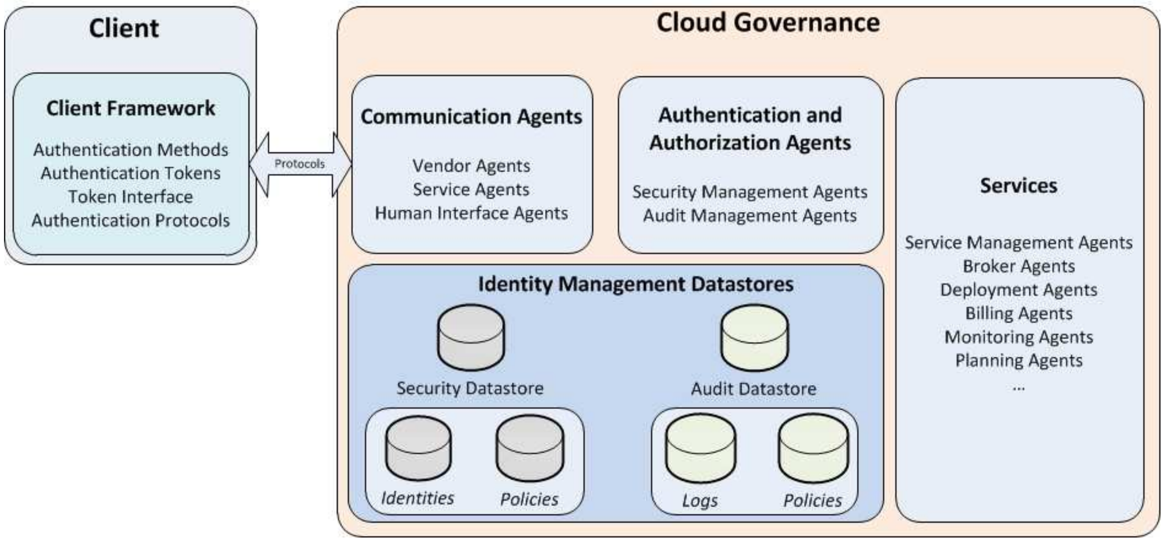

Fig. 4.1: Identity management architecture

- *Human Interface Agent* – is in charge of communicating with human clients. Its role is to enable human interaction with the system, both administrators an users needed for decision/expertise in certain tasks.

The *Authentication and Authorization Agents* handle identity related information like retrieving, generating, logging. They use the *Identity Management Datastores* through two agents:

- *Security Management Agent* – is in charge of validating different types of authentication credentials using information found in the *Security Datastores*. *Communication Agents* forward authentication credentials for validation and receive back authentication tokens that delegate the external party's authority to the agent to which it is connected. *Services* that are offered by cloud governance communicate with this agent in order to validate tokens and receive authorization for the requested actions.
- *Audit Management Agent* – receives and stores all user actions in the system like logins, actions etc. It also monitors them, and, based on policies, can trigger notifications for certain situations.

Services represent the agents performing different actions within the cloud governance solution. They require strong identity authentication and authorization in order to restrict unsanctioned actions by unknown entities.

Identity Management Datastores are distributed databases containing sensitive information about user identities, policies and logs. There are two main datastores:

- *Security Datastore* – stores sensitive user information like identity, credentials (passwords, tokens) as well as policies (permissions). Information related to service instances, cloud providers are also held here.
- *Audit Datastore* – stores information related to user/entity events within the solution like login/logout, service publishing etc., as well as policies which act as rules that, when met, trigger notifications within the system.

4.1. The Security Datastore. Security is a major concern for the users in a cloud environment, being the most important barrier to widespread adoption of cloud computing by the Small and Medium Enterprises. The presented Cloud Governance system meets these security requirements by implementing a complex security mechanism. The access to the services inside the Cloud Governance system is made full or in a granular way and must be performed in a secured manner, only authorized entities having the right to use them. The access rights are better implemented through various security policies that describe the interactions between the services and consumers.

Security Datastore is responsible with holding the credentials for all the participants in the Cloud Governance system, being them service providers or customers, together with their corresponding security policies. In order to access a certain service, the user must be first authenticated by comparing the provided credentials

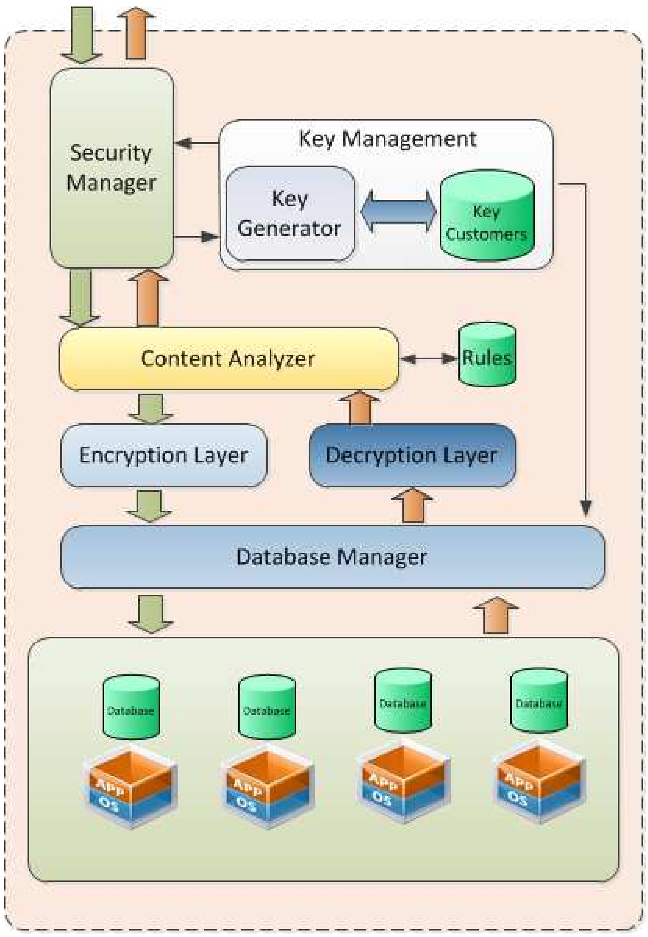

Fig. 4.2: Security Datastore

with the ones stored in the Security Datastore and then, a security token is generated to be used in further services access operations.

The Security Datastore, due its critical importance and extremely sensitive character of the stored data, must be hosted locally inside the Cloud Governance provider's location and not in the Cloud. All the information is encrypted using the AES-256 encryption protocol, and every entity has its own encryption key. More than this, sensitive information related to credit cards or other certificates is split and replicated across many distinct databases hosted on separate machines and recomposed at request, in order to protect the data from possible attacks or accidents. In case of such an incident, not all the records are stolen or lost.

4.2. The Audit Datastore. The mOSAIC's Cloud Agency component performs monitoring tasks for the infrastructure resources negotiated with the cloud providers, but this is not enough in the Cloud Governance context because the registered services need to be also the subject of monitoring in order to raise alerts when the Service Level Agreements (SLAs) are broken or other unpredictable events occurs. The Cloud Governance system offers this upper level monitoring through the Audit and Monitoring Management agent. The source of audit and monitoring data generates a big amount of data, in chunks of various sizes, depending the format exposed by the particular services. This data, once collected, is then refined and aggregated by specialized background processes, and moved in other databases in order to offer better visibility.

It is very important that this information won't be lost due the limited bandwidth to the database so the datastore must be highly writable, which is achieved by a key-value store. On the other hand, the key-value stores are not offering a search mechanism, which is important when data analysis must be performed, so an additional relational database is added in the datastore, to keep the searchable meta-information together

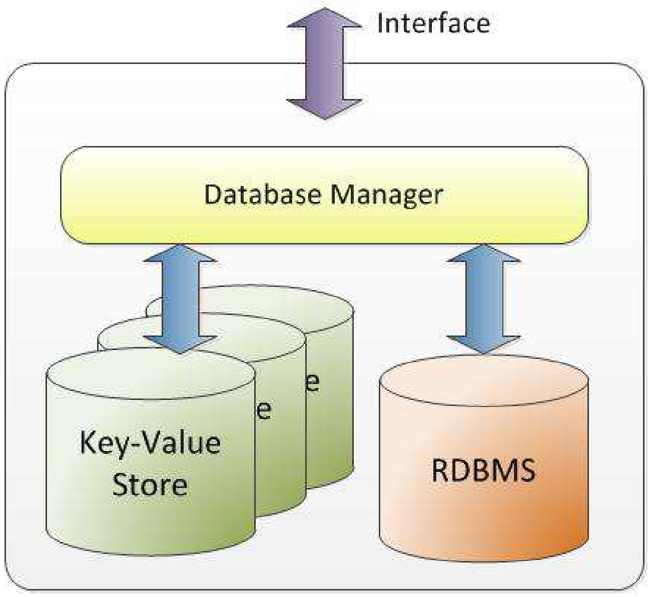

Fig. 4.3: Audit Datastore

with pointers to the keys in the key-value store. The correlation between these databases is made through a specialized software layer called Database Manager, which also implements a transactional mechanism to prevent the de-synchronizations.

5. The Services Datastore. This is the most complex datastore related to the Cloud Governance storage system and its purpose is to persist all the services existent inside the framework, offering in the same time the possibilities for the service providers to publish their final works and for the consumers to discover the services

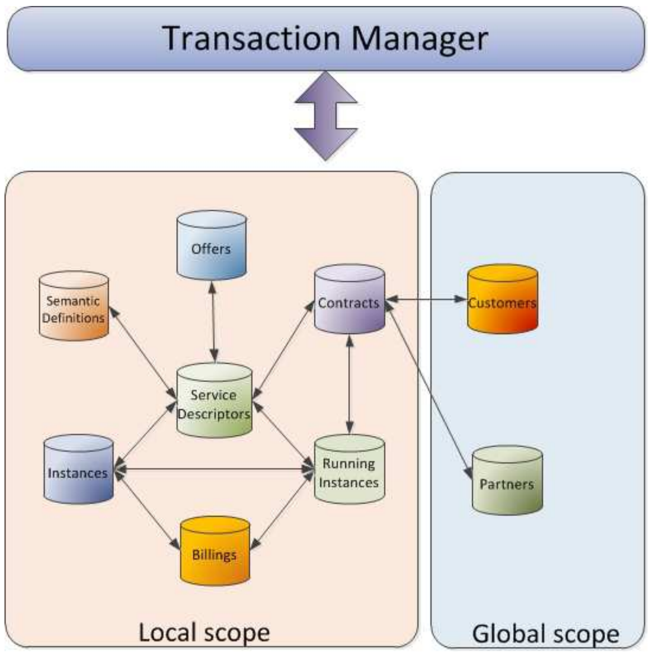

Fig. 5.1: Services Datastore

that best fit their needs and expectations.

The importance of a Service Datastore was emphasized in different white papers, including DMTF's Architecture for Cloud Management document [13], or the ITIL v3 specification, where the Service Catalogue was defined as an entity which "*provides a central source of information on the IT services delivered to the business by the service provider organization, ensuring that business areas can view an accurate, consistent picture of the IT services available, their details and status.*" [6].

5.1. Services Datastore components. The information related to the services is very heterogeneous, part of the data having a well defined structure, while other data being unstructured. All this information must be put in place and coordinated in order to offer a whole vision about the entire system, making possible the smooth functionality of the cloud governance process. In order to simplify the storage system but in the same time to offer a better performance, every kind of information is put in relation with a specialized database. These storage components have a local scope, being visible and accessible only from inside the Services Datastore.

Service Descriptors component contains information related to the services installed in the system. This is the core of the Services Datastore since it contains the description of the services in terms of functional (e.g. QoS, SLA) and non-functional parameters (e.g. Name, Creator, Version, Publisher, etc).

Semantic Definitions component contains information about the existing services, described from a semantic point of view. This information is used in the service discovery process and put in relation with the Service Descriptors which offers a syntactical characterization of the service.

Offers database is responsible with the service offers persistence. Every service must have an offer which better describes its capabilities together with the corresponding price.

Contracts database will hold the contracts resulted from a negotiation or contracting phase between the service provider and service consumer.

Instance Sets and Running Instances databases hold information about the existent services in the system. While Instance Sets contain information about all the service instances in the Cloud Governance, the Running Instances will keep only the services that are executing at a given time.

Billing database is crucial for the service invoicing process, since it keeps all the information generated as a result of service usage.

Complementary to these specialized databases, there are another two important databases called Customers and Partners which maintain relationships with the entities that are using the services and have a global scope visibility, being accessible from all the Cloud Governance components.

Because the Service Datastore is a mesh built from different databases of different types, it is necessary to have a component that is responsible for assuring the atomicity of the operations inside the service publishing use case, the consistency of the data over all the databases, the isolation of the operations and also the durability of the data over the entire datastore. Basically there is a need for a Transaction Manager (TM) that will assure the ACID properties of the Services Datastore as a whole. Due to the distributed character of the Services Datasore, the TM component must deal with distributed transactions which have to assure that all the databases are correctly changed after the operations involving the database modifications. In case of a failure occurred to one or more individual databases, all the changes are rolled back and the Service Datastore is brought to its initial state, in the pre-transactional state. The different types of the databases composing Services Datastore lead to the use of asynchronous transactions, the necessary time to perform the operations over the databases can vary based on the location of the storage, if it is installed locally or remotely, into the cloud and also depending of the amount of data to be written. The TM component is notified by the individual databases about the result of the operations and the transaction is committed only if all the individual database components have successfully performed their changes. If one or more operations fail, the changes are reverted to the initial state. The TM component is based on the X/Open Distributed Transaction Processing [1] specification and implements the two-phase commit protocol (2PC), which ensures that all the individual databases either committed the writes, either rolled back the changes to the pre-transactional state.

5.2. Services Datastore Use Cases. From the very beginning to the end of its existence during the lifecycle a service goes through multiple phases, having many states and interacting with many databases in order to achieve well established goals. In close relation with the Services Datastore, several relevant use cases, associated to lifecycle activities can be described in order to emphasize the interactions that could be considered for this datastore.

These use cases, rather simpler, where selected in order to emphasize the existent and necessary interactions

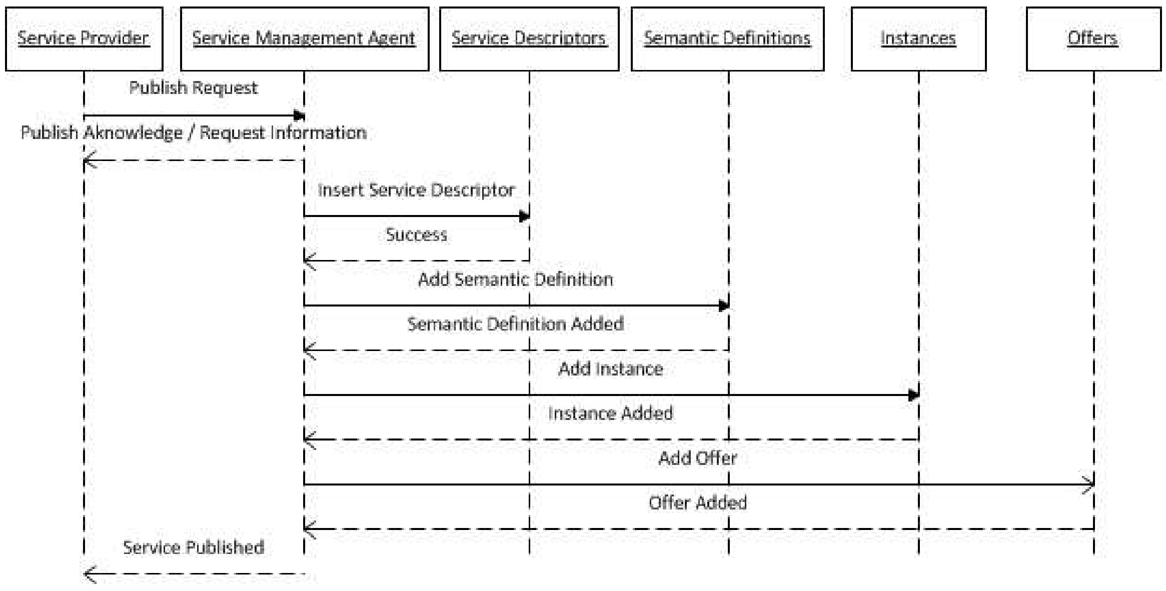

Fig. 5.2: Publishing a service

between the specialized databases composing the Services Datastore. The accent was placed on simplicity, just to keep the discussion at the datastores level and not at the highest level concerning cloud management and cloud governance operations.

5.2.1. Publishing a Service. This is a high level use case and it was selected to emphasize the interactions among some of the Services Datastore components and to reveal the write-intensive character of the operations, being in the same time the first contact between a service provider and the Cloud Governance environment.

In order to be part of the Cloud Governance Authority, a service must conform with a publishing interface which establishes the format and the content of the data that must be provided at the design time by the service creators, like the description of its characteristics, what does it offer, dependencies and price. It is essential that this data exists at the beginning of the publish operation, otherwise the service publishing will fail.

Figure 5.2 reveals the simplified steps followed by the framework to publish a service. All this information is divided, in order to be persisted, in various components of the Services Datastore. The service providers make a request for publishing a service which is received by the Service Management Agent, the entity in charge with the interface to the Services Datastore. The details of this use case together with the interacting actors are presented in Table 5.1 where also information related to the fail conditions are presented.

All the operations over the Services Datastore are performed through the Transaction Manager (TM) component which assures the consistency and integrity of all the composing databases.

When a publish request is received by the Service Management, it will be forwarded to the TM which will assign a transaction identifier for all the operation inside that service publishing. Every operation that changes the content of one of the composing databases must be communicated to the TM which will decide to rollback the entire transaction in case of an operation failure.

5.2.2. Discovering and Contracting a Service. This is a high level use case that shows various interactions between the Services Datastore components with the accent on read-write operations.

This is the first interaction of a service consumer with the Cloud Governance framework. Usually a consumer is interested to find a specific service that meets some characteristics and involving certain constraints related to running costs. Figure 5.3 presents the simplified steps followed by the system to allow a consumer to discover and contract a service. Table 5.2 details all the steps followed in order to discover a service and contract it by a service consumer.

5.2.3. Instantiation and Commissioning a Service. Another consumer interaction with the Cloud Governance system is the instantiation and commissioning phase, as identified in Figure 5.4, in which a service

Use Case: Publishing a service	
Description	
Description and Goal	This use case refers to the service publishing operation, having as result the service available for the consumers over the Cloud Governance environment
Dependencies	None
Actors	Service Provider, Service Management Agent, Service Descriptor, Semantic Descriptor, Offers, Instances
Assumptions and Triggering Events	
Preconditions	All the required information related to the service being published must exist
End conditions	
Success end condition	All the service related information is in the specialized databases, the service is ready for discovery phase
Failure condition	One or more databases signals errors during the change process, the entire distributed transaction is rolled back
Main success scenario	
Normal flow	1. The Service Provider issues a Publish request to the Service Management component, signalling that it wants to publish a service 2. The Service Management responds with a Publish Acknowledge which is a request for the information necessary for a service to be published 3. The Service Management Agent passes the service functional and non functional information to a Broker Agent which is in charge to insert it in the Service Descriptors database. 4. The Service Descriptor reports the result of the operation to the Service Management 5. The semantic definition is passed to the Broker Agent stored in the Semantic Definitions database 6. The Semantic Definitions reports the result of the operation to the Service Management 7. The Service Management passes the information related to the service instance and its dependencies to a Broker Agent which will add them in the Instances database 8. The Instances database reports the result to the Service Management component 9. The information related to the service capabilities and prices is passed to the Broker Agent which adds them in the Offers database. 10. The Offers component reports the result of the operation to the Service Management 11. The Service Management centralizes all the partial results from the Service Datastore components and if all of them are successful, the transaction is committed and the Service Provider is informed that its service was published.

Table 5.1: Publishing a Service Use Case

is launched in execution. This is a high level use case, relevant for the interaction between the components of the Services Datastore, focusing on read-write operations. The details of the use case are revealed in Table 5.3 which presents also all the actors involved in these operation together with the pessimistic scenarios in case of failure.

Use Case: Discovering and Contracting of a Service	
Description	
Description/Goal	This use case refers to the service discovery and contracting operation, having as result the service available to be run in the Cloud Governance environment
Dependencies	Service Publishing completed
Actors	Service Consumer, Service Management Agent, Semantic Definitions, Offers, Contracts
Assumptions and Triggering Events	
Preconditions	The service must be successfully published in a previous step
Postconditions	
Success end condition	The service is successfully discovered, contracted and its instance is ready to be lunched in execution
Failure condition	One or more databases signals errors during the change process, the entire distributed transaction is rolled back
Main success scenario	
Normal flow	1. The Service Consumer makes a request for a specific service to the Services Management with the information about the type of the requested service. 2. The request is transformed by the Service Management into a SPARQL query and passed to a Semantic Definitions to be interrogated 3. The Semantic Definition returns the result of the query to the Service Management 4. The Service Management component passes the result back to the Service Consumer entity which will decide if the service meets its expectations and if further actions will follow. 5. If the Service Consumer entity decides that the discovered service meets its expectation, it informs the Service Management that it is willing for the next step 6. The Service Management queries the Service Descriptor database following the criteria established in the previous step to retrieve the syntactic components of the service 7. The syntactic information is returned to the Service Management which start to compose a local image of the service 8. The Service Management issues a query to the Offers database to retrieve the price of the service 9. The result of the interrogation is returned to the Service Management 10. Service Management passes the result to the Service Consumer, informing it about the price of the service 11. Service Consumer decides if agrees with the service's price and if yes, it sends to the Service Management an Offer Accepted message 12. Service Managements issues an action, which is in fact a modification query, to the Contracts database having a value of contract signature 13. The Contracts database sends back to the Services Management a Service Contracted message 14. The Service Management then informs the Service Consumer about the signed contract.

Table 5.2: Discovering and Contracting a Service Use Case

Use Case: Instantiation and Commissioning a Service	
Description	
Description and Goal	This use case refers to the instantiation and commissioning of a service, having as result the service and executed inside the Cloud Governance environment
Dependencies	None
Actors	Service Consumer, Service Management Agent, Security Management Agent, Security Datastore, Instances, Running Instances, Audit and Billing Management, Billings
Assumptions and Triggering Events	
Preconditions	The service being instantiated must be already published
End conditions	
Success end condition	The service is instantiated and executed, billing information is generated
Failure condition	One or more databases signals errors during the change process, the entire distributed transaction is rolled back
Main success scenario	
Normal flow	1. The Service Consumer issues an Instantiation request to the Service Management component, signalling that it wants to instantiate a service 2. The Service Management passes the request to the Security Management Agent which is in charge with the authorization and authentication operations 3. The Security Management interrogates Security Datastore about the identity of the requester and if successful, it generates the security token that will be attached to all the resource requests 4. The security token is returned by the Security Datastore to the Security Management 5. The Security Management returns the identity and security token back to the Services Management 6. The Services Management issues a query in the Instances database, looking for the specified service. 7. If the specified service is found, it is returned back to the Service Management 8. Having the identity confirmed, a valid security token and an existent service, Service Management is now able to instantiate the service and to issue a query that adds the new running service in the Running Instances database. 9. The result about the new entry in the Running Instances is returned to the Services Management 10. Services Management notifies the Audit and Billing Management component that a new service was started, to produce the information necessary for invoicing the customer. 11. The Audit and Billing Management start to generate billing information and sends it to Billings database. 12. The Billing component responds to the Audit and Billing Management about the result of the operations 13. The Audit and Billing Management informs Services Management that the billing started successfully 14. The Service Management notifies the Service Consumer about teh success of the Instantiation and Commissioning operation

Table 5.3: Instantiation and Commissioning a Service Use Case

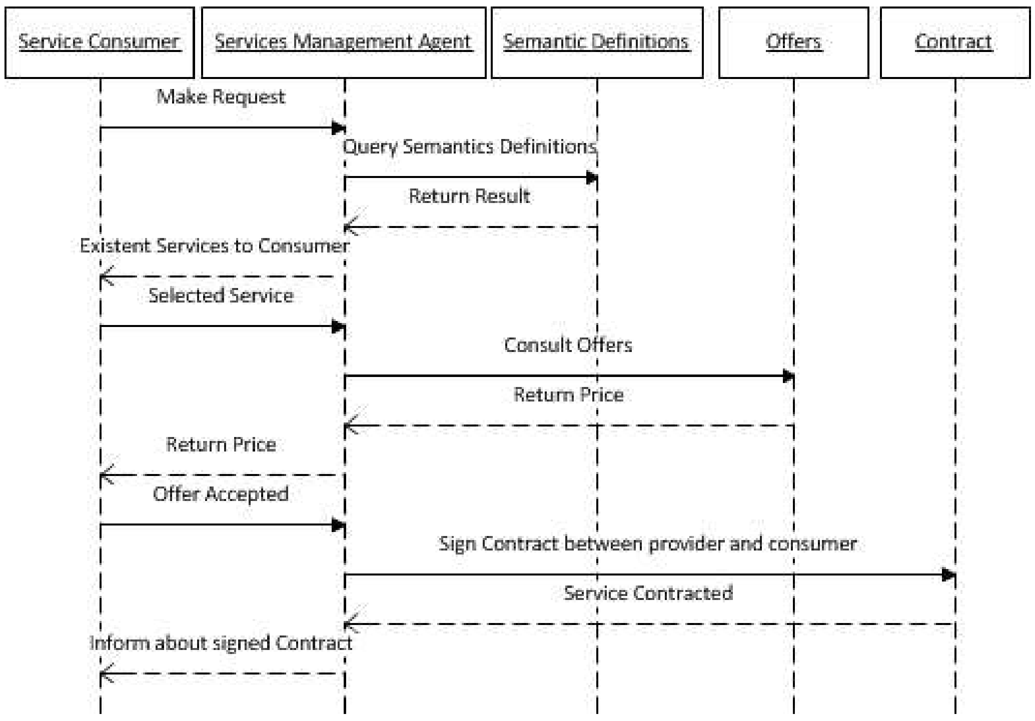

Fig. 5.3: Discovery and Contracting a Service

6. Conclusions and future work. Cloud computing is an evolutionary step in the IT domain, that promises new business and economic models which can be embraced by the SMEs in order perform cost savings and in the same to collaborate in offering complex solutions on an emergent and dynamic market. However, to sustain this virtual collaboration, sophisticated Cloud Governance frameworks that solve the cloud management issues, together with authentication, security, billing and more must be developed. Some companies already integrated cloud governance functionalities in their products, but they do not offer a complete cloud governance solution.

This paper addressed one fundamental aspect of cloud governance: the storage system with its specialized datastores and a series of use cases that emphasize their interaction. The principal existent solutions related to cloud management and cloud governance were identified and also the main database types used in cloud. The architecture of the proposed Cloud Governance system was revealed and the subsystem composing the datastores were detailed. This research conducted in this paper will have fundamental importance in building a working prototype.

Acknowledgments. This work was partially supported by the grant of the European Commission FP7-ICT-2009-5-256910 (mOSAIC), and Romanian national grant PN-II-ID-PCE-2011-3-0260 (AMICAS). The views expressed in this paper do not necessarily reflect those of the corresponding projects' consortium members.

This article is also partially a result of the project "Creşterea calităţii şi a competitivităţii cercetării doctorale prin acordarea de burse"[10]. This project is co-funded by the European Social Fund through The Sectorial Operational Programme for Human Resources Development 2007-2013, coordinated by the West University of Timisoara in partnership with the University of Craiova and Fraunhofer Institute for Integrated Systems and Device Technology - Fraunhofer IISB.

REFERENCES

[10]POSDRU/6/1.5/S/14 Increase the attractiveness, quality and efficiency of university doctoral studies by doctoral scholarships;

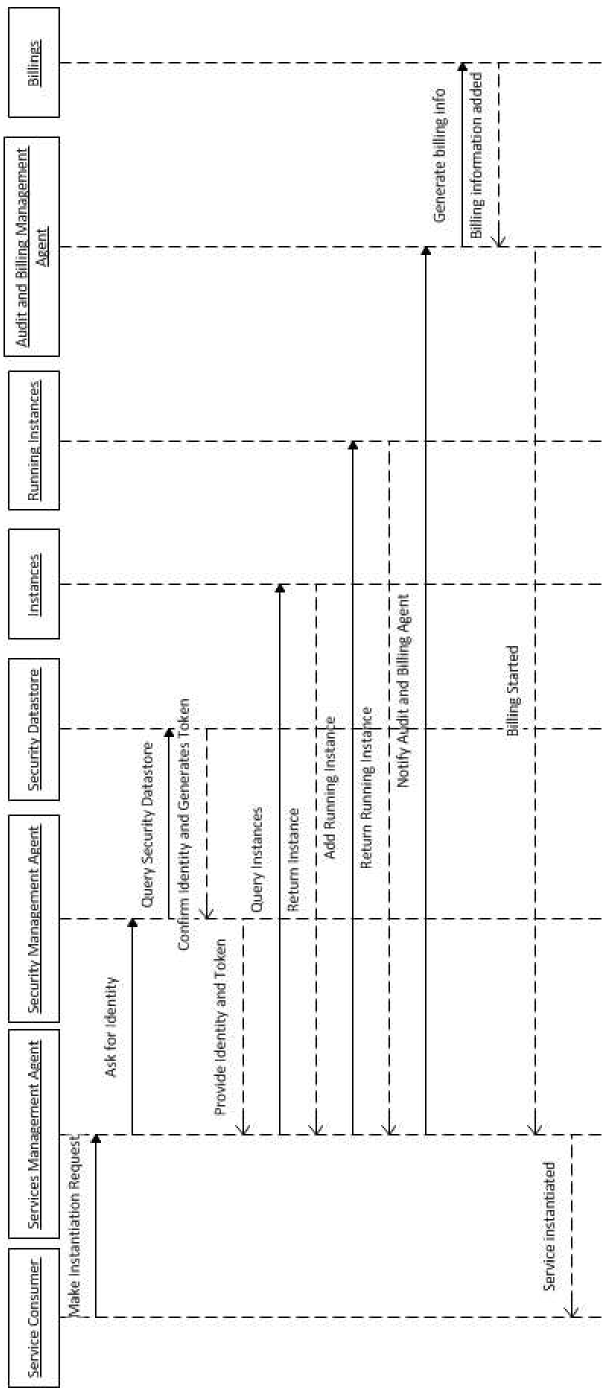

Fig. 5.4: Instantiation and Commissioning a service

[1] *Distributed transactions processing: the xa specifications* `http://pubs.opengroup.org/onlinepubs/009680699/toc.pdf`, November 1998.

[2] *Apache CouchDB*. `http://http://couchdb.apache.org/`, October 2012.

[3] S. Bennett, T. Erl, C. Gee, R. Laird, A. T. Manes, R. Schneider, L. Shuster, A. Tost, and C. Venable, *SOA Governance: Governing Shared Services On-Premise & in the Cloud*, Prentice Hall/PearsonPTR, 2011.

[4] E. A. Brewer, *Towards robust distributed systems.* `http://www.cs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf`, July 2000.

[5] CA Technolofies, *Cloud Management.* http://www.ca.com/us/cloud-platform.aspx, August 2010.

[6] A. Cartlidge, A. Hanna, C. Rudd, I. Macfarlane, J. Windebank, and S. Rance, *An introductory overview of ITIL® v3.* http://www.itilbookshop.org/Documents/an_introductory_overview_of_itil_v3[0].pdf, 2007.

[7] T. Cecere, *Five steps to creating a governance framework for cloud security.* Cloud Computing Journal http://cloudcomputing.sys-con.com/node/2073041, November 2011.

[8] Cloud Computing Use Cases Group, *Cloud computing use cases white paper.* http://opencloudmanifesto.org/Cloud_Computing_Use_Cases_Whitepaper-4_0.pdf, July 2010.

[9] A. Copie, T.-F. Fortis, V. I. Munteanu, and V. Negru, *Service datastores in cloud governance*, in ISPA, IEEE, 2012, pp. 473–478.

[10] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, *Dynamo: Amazon's highly available key-value store*, SIGOPS Oper. Syst. Rev., 41 (2007), pp. 205–220.

[11] B. Di Martino, D. Petcu, R. Cossu, P. Goncalves, T. Máhr, and M. Loichate, *Building a mosaic of clouds*, in Euro-Par 2010 Parallel Processing Workshops, M. Guarracino, F. d. r. Vivien, J. Traff, M. Cannatoro, M. Danelutto, A. Hast, F. Perla, A. Knapfer, B. Di Martino, and M. Alexander, eds., vol. 6586 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2011, pp. 571–578.

[12] DMTF, *Architecture for Managing Clouds.* http://dmtf.org/sites/default/files/standards/documents/DSP-IS0102_1.0.0.pdf, June 2010.

[13] DMTF, *Use Cases and Interactions for Managing Clouds.* http://www.dmtf.org/sites/default/files/standards/documents/DSP-IS0103_1.0.0.pdf, June 2010.

[14] enStratus, *Enterprise Cloud Governance White Paper.* http://www.enstratus.com/page/1/ecg-wp-track.jsp, August 2010.

[15] C. Escapa, *Cloud Management Guide.* White paper http://www.abiquo.com/files/white_paper_cloud_management_guide.pdf, August 2010.

[16] T.-F. Fortis, V. I. Munteanu, and V. Negru, *Steps towards cloud governance. a survey*, in ITI, 2012, pp. 29–34.

[17] S. Gilbert and N. Lynch, *Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services*, SIGACT News, 33 (2002), pp. 51–59.

[18] C. Gong, J. Liu, Q. Zhang, H. Chen, and Z. Gong, *The characteristics of cloud computing*, in ICPP Workshops, W.-C. Lee and X. Yuan, eds., IEEE Computer Society, 2010, pp. 275–279.

[19] G. Gruman, *Integration issues may hinder saas adoption.* CIO Update http://www.cioupdate.com/trends/article.php/3695096/Integration-Issues-May-Hinder-SaaS-Adoption.htm, August 2007.

[20] F. Liu, J. Tong, J. Mao, R. Bohn, J. Messina, L. Badger, and D. Leaf, *NIST cloud computing reference architecture.* http://collaborate.nist.gov/twiki-cloud-computing/pub/CloudComputing/ReferenceArchitectureTaxonomy/NIST_SP_500-292_-_090611.pdf, September 2011.

[21] P. Mell and T. Grance, *The NIST Definition of Cloud Computing.* White paper http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf, September 2011.

[22] mOSAIC Consortium, *The mOSAIC project*, October 2012.

[23] F. Moscato, R. Aversa, B. D. Martino, T.-F. Fortiş, and V. Munteanu, *An analysis of mosaic ontology for cloud resources annotation.*, in FedCSIS, M. Ganzha, L. A. Maciaszek, and M. Paprzycki, eds., 2011, pp. 973–980.

[24] F. Moscato, R. Aversa, B. D. Martino, D. Petcu, M. Rak, and S. Venticinque, *Cloud computing: methodology, system, and applications*, CRC, Taylor & Francis group, 2011, ch. An Ontology for the Cloud in mOSAIC.

[25] A. Mulholland, J. Pyke, and P. Fingar, *Enterprise Cloud Computing: A Strategy Guide for Business and Technology Leaders*, Meghan-Kiffer Press, Tampa, FL, USA, 2010.

[26] V. I. Munteanu, T.-F. Fortis, and A. Copie, *Building a cloud governance bus*, International Journal of Computers, Communications and Control, 7 (2012), pp. 888–894.

[27] NIST, *Cloud architecture reference models: A survey.* http://collaborate.nist.gov/twiki-cloud-computing/pub/CloudComputing/Meeting4AReferenceArchtecture013111/NIST_CCRATWG_004v2_ExistentReferenceModels_01182011.pdf, January 2011.

[28] C. Ruz, F. Baude, B. Sauvan, A. Mos, and A. Boulze, *Flexible SOA lifecycle on the cloud using SCA*, in Enterprise Distributed Object Computing Conference Workshops (EDOCW), 2011 15th IEEE International, 29 2011-sept. 2 2011, pp. 275 –282.

[29] K. Tang, J. M. Zhang, and C. H. Feng, *Application centric lifecycle framework in cloud*, E-Business Engineering, IEEE International Conference on, 0 (2011), pp. 329–334.

[30] T. Tung, *Defining a cloud reference model*, in Cluster Computing and the Grid, IEEE International Symposium on, IEEE Computer Society, 2011, pp. 598–603.

[31] S. Venticinque, R. Aversa, B. Di Martino, M. Rak, and D. Petcu, *A cloud agency for SLA negotiation and management*, in Proceedings of the 2010 conference on Parallel processing, Euro-Par 2010, Berlin, Heidelberg, 2011, Springer-Verlag, pp. 587–594.

[32] S. Venticinque, R. Aversa, B. D. Martino, and D. Petcu, *Agent based Cloud Provisioning and Management - Design and Prototypal Implementation*, in CLOSER 2010, 2011, pp. 184–191.

[33] C. Weinhardt, A. Anandasivam, B. Blau, N. Borissov, T. Meinl, W. Michalk, and J. Stösser, *Cloud Computing– A Classification, Business Models, and Research Directions*, Business and Information Systems Engineering (BISE), 1 (2009), pp. 391–399. ISSN: 1867-0202.

[34] C. Weinhardt, A. Anandasivam, B. Blau, and J. Stösser, *Business Models in the Service World*, IEEE IT Professional, Special Issue on Cloud Computing, 11 (2009), pp. 28–33. ISSN: 1520-9202.

[35] WSO2, *Governance Registry.* White paper http://wso2.com/products/governance-registry/, August 2010.

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