Preprint: Karuna P Joshi, Tim Finin, Yelena Yesha, Anupam Joshi, Navid Golpayegani and Nabil R. Adam, A Policy-based Approach to Smart Cloud Services, Annual Service Research and Innovation Institute Global Conference, San Jose, July 2012.

A Policy-based Approach to Smart Cloud Services

Karuna P Joshi, Tim Finin, Yelena Yesha,
Anupam Joshi, Navid Golpayegani
Computer Science and Electrical Engineering
University of Maryland, Baltimore County
Baltimore, MD, USA
{kjoshi1, finin, yeyesha, joshi, golpa1}@umbc.edu

Nabil R. Adam
CIMIC
Rutgers University
Newark, NJ, USA
adam@adam.rutgers.edu

***Abstract*— Virtualized service models are now emerging and redefining the way information technology is delivered. Managing these services efficiently over the cloud is an open challenge. We are developing a policy-based integrated framework for automating acquisition and consumption of Cloud services. We have developed a tool, Smart Cloud Services tool, to automatically discover, negotiate and consume services from the cloud for a specific use case described by NIST around storage services. This tool incorporates NIST specific definitions for cloud services. The tool makes use of Semantic Web technologies including SPARQL, RDF and OWL to represent and reason about services and service requirements. It is built upon the service lifecycle ontology that we have developed. In this paper, we describe this Smart Cloud Services tool in detail along with the technical challenges we faced in developing it and identify limitations in the current open source cloud computing software.**

Keywords-cloud services; cloud computing; policies; semantic web

I. INTRODUCTION

Organizations are increasingly adopting an Information Technology (IT) delivery model where components of IT like software, hardware or network bandwidth can be purchased as services from providers based anywhere in the world. Typically the service is hosted on a Cloud and is delivered to the organization via the Internet or mobile devices. The service is acquired on an as needed basis and can be described as service on demand. In such an environment, multiple providers often collaborate to create a single service for an organization. In some cases, businesses utilize multiple service providers to mitigate risks that may be associated with a single provider. In others, a business may use a single provider who in turn utilizes the services of other providers. In either case, the delivery of IT service is moving away from a single provider mode, and is increasingly based on the composition of multiple services and assets (technological, human, or process) that may be supplied by one or more providers distributed across the network – in the cloud. Moreover, a single service can be a part of many composite services as needed. The service, in effect, is virtualized on the cloud. This is becoming the preferred method to deliver services ranging from helpdesk and back-office functions to *Infrastructure as a Service* (IaaS). The virtualized model of service delivery also extends to IT Enabled Services (ITeS), which typically include a large human element.

A key barrier preventing organizations from successfully using services on the cloud is that the they have complex internal policies, as well as legal and statutory constraints that require compliance. Such policies are today enforced on internal resources controlled by the organization. When acquiring remote services, it requires significant human intervention and negotiation -- people have to check whether a provider's service attributes ensure compliance with their organization's constraints. This can get very complex if the provider is composing services, some of which it gets from other providers. A related issue is the lack of an integrated methodology for service creation and deployment that provides a holistic view of the service lifecycle on a cloud. We are developing a methodology [6] to address the lifecycle issue for virtualized services delivered from the cloud and describe it briefly in section III. This lifecycle provides ontologies to describe services and their attributes. In particular, we used semantically rich descriptions of the requirements, constraints, and capabilities that are needed at each phase of the lifecycle. Policies can be described using the same ontology terms so that compliance checks can be automated. This methodology is complementary to previous work on ontologies, e.g., OWL-S [10], for service descriptions in that it is focused on automating processes needed to procure services on the cloud. The methodology will enable practitioners to plan, create and deploy virtualized services successfully.

We have developed and implemented a cloud storage service prototype to demonstrate and evaluate our methodology. The prototype allows cloud consumers to discover and acquire disk storage on the cloud by specifying the service constraints, security policies and compliance policies via a simple user interface. This prototype was developed as part of our collaboration with National Institute of Standards and Technology (NIST). We used W3C standard Semantic Web technologies, such as OWL [12], RDF [9], and SPARQL [18], to develop our prototype system since they enable us to build the vocabulary (or ontology) of our service lifecycle using standardized languages that support our design requirements, which

include interoperability, sound semantics, Web integration, and the availability of tools and system components.

Our most fundamental requirement is for a representation that supports interoperability at both the syntactic and semantic levels. The OWL [12] language has a well-defined semantics that is grounded in first order logic and model theory. This allows programs to draw inferences from OWL expressions with the assurance that the subsequent interpretation is sound. An important advantage for OWL over many other knowledge-based systems languages is that there are well defined subsets (or profiles, as they are called in OWL 2) that guarantee sound and complete reasoning with various levels of completely (e.g., N2ExpTime for OWL 2 DL). Moreover, there are also profiles that are tuned to work well with popular implementation technologies, e.g., OWL QL for databases and OWL RL for rule-based systems.

A second design requirement is for a language that is designed to integrate well with the Web, which has become the dominant technology for today's distributed information systems. OWL is built on basic Web standards and protocols and is evolving to remain compatible with them. It is possible to embed RDF and OWL knowledge in HTML pages and several search engines (including Google) will find and process some embedded RDF. RDF is also compatible with Microdata, a Web Hypertext Application Technology Working Group HTML specification that is used to nest semantic statements within existing content on web pages. Microdata has been adopted by Schema.org, a collaboration by Google, Microsoft, and Yahoo!, and has been used to define a number of basic ontologies that are being supported by search engines.

Finally, there are a wide variety of both commercial and open sourced tools that support Semantic Web languages and systems including knowledge base editors, reasoners, triple stores, SPARQL query engines (including some that support federated queries), ontology mapping, etc. Several database vendors, including Oracle and IBM, have sophisticated support for representing RDF and OWL, including reasoning.

II. Related Work

Since cloud computing is a nascent field, there is lack of standardization and there seems to be a need to clearly define its key elements. NIST has released a special publication 800-145 [13] defining cloud computing as a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. One of the key characteristics identified by NIST is that a cloud service should have the capability of on-demand self-service whereby a consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider. This capability of automatically acquiring a service is currently either missing, or very limited, in most cloud based services. Our methodology aims to make it possible to automatically discover, negotiate/acquire and consume cloud based services.

In addition to the standard definition of Cloud Computing, NIST has also released the Cloud Computing Reference Architecture document [14] that describes a reference architecture for cloud computing and also the key roles and responsibilities of stakeholders. The authors of this paper were part of the NIST cloud computing reference architecture and taxonomy working group that participated in developing the standard. Our ontology, described in the next section, makes use of the NIST cloud computing standards..

Current research on cloud or web services so far has been limited to exploring a single aspect of the lifecycle such as service discovery, service composition, or service quality. There is no integrated methodology for the entire service lifecycle - covering service planning, development and deployment in the Cloud. In addition, most of the work is limited to the software component of the service and does not cover the service processes or human agents which are a critical component of IT Services.

Papazoglou and Heuvel [16] have proposed a methodology for developing and deploying web services using service oriented architecture (SOA). Their approach, however, is limited to the creation and deployment of web services and does not account for virtualized environment where services are composed on demand. Providers may need to combine their services with other resources or providers' services to meet consumer needs. Other methodologies, like that proposed by Bianchini et al. [1], do not provide this flexibility and are limited to cases where a single service provider provides one service. Zeng et al. [28] address the quality based selection of composite services via a global planning approach but do not cover the human factors in quality metrics used for selecting the components. Maximilien and Singh [11] propose an ontology to capture quality of a web service so that quality attributes can be used while selecting a service. While their ontology can serve as a key building block in our system, it is limited by the fact that it considers single web services, rather than service compositions.

Black et al. [2] have proposed an integrated model for IT service management. Their model is limited to managing the service from the service provider's perspective. Paurobally et al. [17] have described a framework for negotiation of web services using the iterated Contract Net Protocol (CNP) [21]. However their implementation is limited to pre-existing web services and doesn't extend to virtualized services that are composed on demand. Our negotiation protocol detailed in next section accounts for the fact that the service will be composed only after the contract/SLA listing the constraints is finalized. GoodRelations [3] is an ontology developed for E-commerce to describe products. While this ontology is

useful for describing service components that already exist on the cloud, it is difficult to describe composite virtualized services being provided by multiple vendors using this ontology. Cardoso et al. [24] have described a Unified Service Description Language (USDL) a specification language to describe services from a business, operational and technical perspective.

The Information Technology Infrastructure Library (ITIL) is a set of concepts and policies for managing IT infrastructure, development and operations that has wide acceptance in the industry. The latest version of ITIL lists policies for managing IT services [23] that cover aspects of service strategy, service design, service transition, service operation and continual service improvement. However, it is limited to interpreting "IT services" as products and applications that are offered by in-house IT department or IT consulting companies to an organization. This framework in its present form does not extend to the service cloud or a virtualized environment that consists of one or more composite services generated on demand.

We use Semantic Web techniques for our services lifecycle and prototype development. The Semantic Web deals primarily with data instead of documents. It enables data to be annotated with machine understandable meta-data, allowing the automation of their retrieval and their usage in correct contexts. Semantic Web technologies include languages such as Resource Description Framework (RDF) [9] and Web Ontology Language (OWL) [12] for defining ontologies and describing meta-data using these ontologies as well as tools for reasoning over these descriptions. These technologies can be used to provide common semantics of Service information and policies enabling all agents who understand basic Semantic Web technologies to communicate and use each other's data and Services effectively. The Ontology Web Language for Services (OWL-S) [10] was developed to provide a vocabulary for describing the properties and capabilities of Web Services in unambiguous, computer-interpretable form. OWL-S allows Service providers or brokers to define their Services based on agreed upon ontologies that describe the functions they provide. We have integrated the OWL-S ontology into our ontology.

SPARQL Protocol and RDF Query Language (SPARQL) is the query language for RDF that has been standardized by W3C [18]. SPARQL can be used to express queries across diverse data sources, whether the data is stored natively as RDF or viewed as RDF via middleware. It has capabilities for querying required and optional graph patterns and their conjunctions and/or disjunctions. SPARQL also supports value testing and altering of results. The results of queries can be results sets or RDF graphs. A SPARQL abstract query is defined in [18] as a tuple (E, DS, R) where E is a SPARQL algebra expression, DS is an RDF Dataset and R is a query form.

A SPARQL endpoint is a conformant SPARQL protocol service as defined in the SPARQL Protocol for RDF (SPROT) specification [22]. It enables users (human or other) to query a knowledge base via the SPARQL language. Results are typically returned in one or more machine-processable formats. Therefore, a SPARQL endpoint is mostly conceived as a machine-friendly interface towards a knowledge base. Both the formulation of the queries and the human-readable presentation of the results should typically be implemented by the calling software, and not be done manually by human users. We have used the Joseki [5] server to simulate the SPARQL endpoint for our tool.

III. Lifecycle of Cloud Services

We have developed a methodology that integrates all the processes and data flows that are needed to automatically acquire, consume, and manage services on the cloud. We divide this IT service lifecycle on a cloud into five phases. In sequential order of execution, they are requirements, discovery, negotiation, composition, and consumption; and are illustrated in figure 1. We have described these phases in detail along with the associated metrics in [6]. We have developed the ontology for the entire lifecycle in OWL 2 DL profile, [7]. This ontology has been used in the development of the Smart Cloud Services tool described in the next section.

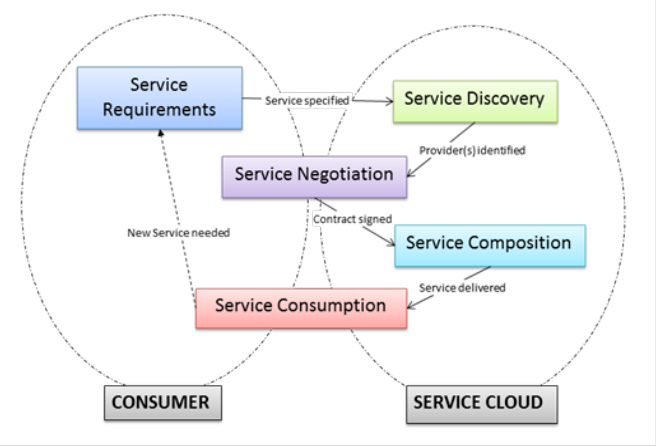

Figure 1: The IT service lifecycle on a virtualized cloud comprises five main phases: requirements, discovery, negotiation, composition and consumption

A. *Service Requirements Phase*

In this phase, the consumer details the technical and functional specifications that a service needs to fulfill along with the organizational policies for the providing agent, data quality policy and security policies for the service. Service compliance policies such as required certifications standards to be adhered to, etc. are also identified. Depending on the service cost and availability, a consumer may be amenable to compromise on the service quality. Functional specification describe in detail what functions/tasks should a service help automate. The technical specifications lay down the hardware, software, application standards, and language

support policies to which a service should adhere. Once the consumers have identified and classified their service needs, they issue a Request for Service (RFS). This request could be made by directly contacting a few service providers for their quotes. Alternatively, consumers can use a service discovery engine or service broker on the cloud to procure the service.

B. Service Discovery Phase

Service providers are discovered by comparing the specifications listed in the RFS. The discovery is constrained by functional and technical attributes defined, and also by the budgetary, security, compliance, data quality, and agent policies of the consumer. While searching the cloud, service brokers can be employed. The broker engine queries the service providers to match the service domain, data type, compliance needs, functional, and technical specifications; and returns the result with the service providers in priority order. Sbodio et al. [20] and Paliwal et al. [25] have presented semantic approaches for service discovery which can be incorporated in our methodology.

One critical part of this phase is service certification, in which the consumers will contact a central registry, such as UDDI [19], to get references for providers that they narrow down to.

If a consumer finds the exact service within their budgets, s/he can begin consuming the service immediately upon payment. However, often the consumer will get a list of providers who will need to compose a service to meet the consumer's specifications. The consumer will then have to begin negotiations with the service providers. Each search result will return the primary provider who will be negotiating with the consumer.

C. Service Negotiation phase

The service negotiation phase covers the discussion and agreement that the service provider and consumer have regarding the service delivered and its acceptance criteria. The service delivered is determined by the specifications laid down in the RFS. Service acceptance is usually guided by the Service Level Agreements (SLA) that the service provider and consumer agree upon. SLAs define the service data, delivery mode, agent details, quality metrics, and cost of the service. While negotiating the service levels with potential service providers, consumers can explicitly specify service quality constraints (data quality, cost, security policies, response time, etc.) that they require.

At times, the service provider will need to combine a set of services or compose a service from various components delivered by distinct service providers in order to meet the consumer's requirements. The negotiation phase also includes the discussions that the main service provider has with the other component providers. When the services are provided by multiple providers (composite service), the primary provider interfacing with the consumer is responsible for composition of the service. The primary provider will also have to negotiate the Quality of Service (QoS) with the secondary service providers to ensure that SLA metrics are met.

The key deliverable of this phase is the service contract between the service consumer and service provider. The SLA is a key part of this service contract and will be used in the subsequent phases to compose and monitor the service. Another deliverable of this phase are the service sub contracts between the service provider and component (or dependent services) providers. The QoS are the essential part of the service sub-contracts and are used in the consumption phase to monitor service performance.

D. Service Composition phase

In this phase one or more services provided by one or more providers are combined and delivered as a single service. Service orchestration determines the sequence of the service components.

E. Service Consumption/Monitoring phase

The service is delivered to the consumer based on the delivery mode (synchronous/asynchronous, real-time, batch mode etc.) agreed upon in the negotiation phase. After the service is delivered to the consumer, payment is made and the consumer then begins consuming the service. In a cloud environment, the service usually resides on remote machines managed by the service providers. The provider is responsible for managing and monitoring the service. In this phase, consumer will require tools that enable service quality monitoring and service termination if needed. This will involve alerts to humans or automatic termination based on policies defined using the quality related ontologies. The Service Monitor measures the service quality and compares it with the quality levels defined in the SLA. This phase spans both the consumer and cloud areas as performance monitoring is a joint responsibility. If the consumer is not satisfied with the service quality, s/he should have the option to terminate the service and stop service payment.

The composite service is made up of human agents providing the service, the service software, and dependent service components. All the three elements, agents, software, and dependent services, must be monitored to manage the overall service quality. For the service, software providers have to track its performance, reliability, assurance, and presentation as they will influence customer's satisfaction rating (CSATs). Since the dependent services/components will be at the backend and will not interface directly with the consumers, the service provider only needs to monitor their performance. We have proposed a framework to manage quality based on fuzzy-logic for such composed services delivered on the cloud in [8].

IV. SMART CLOUD SERVICES TOOL

In this section we describe the prototype that we have constructed in collaboration with NIST for automatically acquiring cloud based services. This tool is based on our

proposed lifecycle that we described in the previous section and uses the ontology that we have developed for the same. It demonstrates the capability that cloud users will have in the future to automatically acquire Cloud services. This tool allows users to define a range of values for their constraints and so accommodates for scenarios where the users may not have finalized their requirements.

For the prototype we considered a simple Storage service, as a representative scenario for Infrastructure as a Service (IaaS), whereby users can store their files/data on the cloud. It consists of a web interface (Figure 2) that enables cloud users to easily define the service policies and constraints by choosing predefined values from dropdown fields. The tool then discovers the services that will match the specified policies. A Cloud-provider end server process interprets the policies specified by the user(s) and establishes Service Level Agreements (SLAs) by the process of negotiation.

Each field in the interface has an associated 'Help' description that describes the attributes and enables users to determine which option to select. The fields in each section are placed in descending order of priority. This tool allows users to define a range of values for their constraints and so accommodates for scenarios where the users may not have finalized their requirements.

A. Prototype Description

We used Semantic Web technologies to build the front end of our prototype as they are platform independent and inter-operable. We used SPARQL, Jena Semantic Web framework [4] and the Joseki software [5], which is an HTTP engine that supports the SPARQL Protocol and the SPARQL RDF Query language, to develop the prototype. After defining our service, we created a SPARQL endpoint using Joseki to simulate a service provider providing the service. Since the Joseki server allows multiple service definitions, we used it to simulate both multiple services provided by the provider as well as multiple instances of a same service. The Joseki service database contained the service description along with the provider policies endpoint. For the cloud-end processes, we used the Eucalyptus Cloud [15] which is an open source cloud platform that we have installed in our research lab. We are using our service lifecycle ontology that we described in the previous section and the OWL-S ontology to develop the tool. In addition to these two ontologies, we also created an OWL ontology to describe the technical and security policies for our prototype.

We have incorporated actual enterprise policies related to data storage and security that are practiced by large organizations. We have used the policies defined in the use case 3.9 [26] identified by the NIST cloud computing initiative. While requesting the storage service, users will specify the four categories of service attributes –core service attributes, data/security policy attributes, compliancy policy attributes and Cloud instance attributes.

1) CORE Attributes

The Core attributes include the mandatory attributes that the service provider should meet. The attributes in this section are listed below.

a) Storage size

This attribute defines the storage size (in Giga Bytes / Tera Bytes units) that the consumer wishes to procure on the cloud. To enable consumers to easily determine their storage needs, the field in the tool consists of a drop down list with a range of values – 'less than 2GB', '2GB to 10GB', '10GB to 50 GB', '50 GB to 100 GB', '100 GB to 1TB' and 'more than 1TB' . We referenced existing cloud storage solutions to determine the storage ranges offered in the market today.

b) Service Cost

This attribute refers to the price consumers are willing to pay for the service. The options listed in this field provide a sample of the pricing models used currently by storage service providers. These can be modified based on service domain. The service cost will vary depending on the other attributes specified by the cloud user and so the service is searched on a range of costs as opposed to a specified price. The ranges provided in the tool are 'Free / no cost', '$0 to $10 per month', '$10.01 to $20 per month' and '$20.01 to $30 per month'. For simplicity we have adopted the subscription based pricing model in the tool.

c) Data Preservation/Backup

This attribute specifies the data backup requirements. The consumers select the Daily or Hot backup options if they want the cloud provider to backup their data often. For instance, if users are constantly updating the files stored on the cloud, then the 'hot backup' option will better suit their needs. The tool cautions consumers that the service cost attribute will be higher if more frequent backup options are selected. The ranges provided in the tool are 'Daily', 'Weekly' and 'Hot backup'.

d) Service Availability

This field specifies the minimum level of service availability consumers expect from the service provider. Consumers are advised to select higher availability options if they expect the service will be heavily used. The tool cautions consumers that the service cost attribute will be higher if service availability is set high. The ranges provided in the tool are '95%', '99%' and '99.9%'.

2) DATA/SECURITY POLICY Attributes

The data and security policy attributes specify the policies of the consumer organization with regards to their data on the cloud. The policies that we selected are specific to the NIST cloud computing User case 3.9 [26]. The priority of

each policy was determined by our NIST collaborators and the fields were positioned in the tool in decreasing order of their priority. The attributes in this section are listed below.

a) User authentication mechanism

NIST issued FIPS (Federal Information Processing Standard - Publication 140-2) is a U.S. government computer security standard used to accredit cryptographic modules. This attribute specifies whether FIPS 140-2 is to be supported by the cloud provider or not.

b) Data Encryption

This attribute specifies if the consumer wants the data to be encrypted when stored on the cloud. The tool cautions consumers that the Cloud providers will charge more for the service if data is stored encrypted.

c) Data Location

This attribute specifies the constraint the consumer may have regarding the location of the cloud. The options available in the tool are 'Anywhere in the world', 'Restricted to US jurisdiction' and 'Restricted to EU jurisdiction'. Restricted to US jurisdiction includes all the locations in the world where US laws specific to cloud computing can be enforced. Restricted to European Union (EU) jurisdiction includes all the locations in the world where EU laws specific to cloud computing can be enforced. For instance, EU mandates that the cloud infrastructure should exist within EU countries.

d) Data Deletion

This attribute helps consumer specify their data deletion policies for the cloud – whether the data is deleted or merely made inaccessible, secure wipe is supported or not. While in the cloud environment it may be difficult to ensure the intended data deletion; adding this policy to the Service SLA will make the cloud provider liable if the deletion method specified is not followed. So the onus to ensure appropriate data deletion procedure will be on the cloud provider and not the cloud user.

e) Virtual Machine (VM) separation

This attribute specifies whether the cloud provider supports separate Virtual Machines to store consumer's data on the cloud. Some organizations may desire separate Virtual Machines on the cloud to ensure more security.

f) Interface for a storage specification.

The consumer can specify in their requirements whether they want SOAP protocol or REST (Representational state transfer) interface support.

3) COMPLIANCE POLICY Attributes

In the tool we have included two compliance policies specified by NIST and majority of the US federal agencies. They are listed below. The compliance policy constraints have lower priority than the data/security policy constraints and hence are relaxed after cloud instance constraints, but before the data/security constraints during service negotiation.

a) Trusted Internet Connection (TIC)

Trusted Internet Connection initiative is mandated in an OMB (Office of Management and Budget) Memorandum (M-08-05) meant to optimize individual external connections, including internet points of presence currently in use by the Federal government of the United States.

b) CC Evaluation Assurance Level (EAL) levels

The Evaluation Assurance Level (EAL1 through EAL7) of an IT product or system is a numerical grade assigned following the completion of a Common Criteria security evaluation, an international standard in effect since 1999. This attribute accepts EAL number from 1 to 7.

4) CLOUD INSTANCE Attributes

If consumers have specific requirements of the Cloud instance, they can specify the RAM size, CPU speed and number of cores dedicated to the requested service. The Cloud Instance policy constraints have the lowest priority and hence are relaxed first during service negotiation.

B. Prototype Workflow

The prototype mirrors the service lifecycle described in section III. After selecting the values of their core attributes, security policies, compliance policies and cloud instance, the consumers can press the 'Request for Service (RFS)' button to generate a RDF document that contains the RFS. Figure 2 illustrates the RFS in a RDF/XML format.

After generating the RFS, users press the 'Discover Services' button to search for services that match the RFS constraints. The tool generates federated SPARQL queries based on the selections on the screen. This query runs across multiple SPARQL endpoints and retrieves a list of matching services residing on that endpoint. If a query matching all the constraints is found, it is displayed on the screen. Otherwise, the user is advised to begin service negotiation by selecting the Negotiation button.

The users press the 'Negotiate and Finalize SLA' button to begin service negotiation. The tool iteratively relaxes the requirements one by one and generates a new SPARQL query to search the service endpoints. The order of constraint relaxation for this prototype was determined by the NIST team that was collaborating with us who specified the priority of each constraint in the RFS. After each constraint relaxation, the tool executes the new SPARQL query to discover services that match the new constraint set. When a service match is found, the tool returns the service details of that service along with a list of constraints not met (illustrated in figures 3, 4 after the reference section). The consumer can finalize the SLA by accepting the service that

best matched the constraints. The final SLA (figure 5) is generated as a RDF file and is in machine readable format.

The user next click the 'Compose Services on the Cloud' button to compose the desired service.

```
@prefix itso: <http://ebiq.org/o/itso/1.0/itso.owl>.
@prefix stg: <http://ebiq.org/o/storage/1.0/storage.owl>.

[a itso:Service_Level_Agreement
 itso:Expected_Begin_Date_of_Service "1-1-2012";
 itso:Service_Cost_Constraint "0";
 itso:Service_Location_constraint "global";
 stg:authentication "FIPS 140 2 supported";
 stg:availability "95";
 stg:backup "Weekly";
 stg:cloud_instance_cores "4";
 stg:cloud_instance_size "1073741824";
 stg:cloud_instance_speed "1GHz";
 stg:datadeletion "data archived";
 stg:Encryption "No Encryption";
 stg:storage "1073741824";
 stg:storage_interface "SOAP WSDL";
 stg:TIC_connection "TIC Compliant";
 stg:VMseparation "VM separation". ]
```

Figure 5: The finalized SLA is stored as a RDF document serialized in the Terse RDF Triple Language (Turtle).

C. *Service Composition and Consumption on the Cloud*

When the user clicks the Compose button, a Virtual Machine is created on the Eucalyptus cloud environment. The finalized SLA is referred to by an automated routine when launching the virtual machine. This URI of the service is then returned to the end user to begin consuming the service. By clicking on the Launch Service button, the consumer is directed to the service URI on Eucalyptus cloud environment.

To interface our tool to a cloud system we chose Eucalyptus [15], an Infrastructure as a Service (IaaS) cloud solution. We chose it since it is open source and is compatible with the Amazon Web Services (AWS) cloud. Smart Cloud services tool and the Eucalyptus cloud were installed on separate machines. Figure 6 shows the layout of the installation. Due to security reasons, the Eucalyptus installation had no direct internet access and no direct access to the tool. The only way to communicate between the two systems was through an intermediate node called Bluegrit which is a 116 core PowerPC cluster managed at UMBC.

The software to manage the interaction between our tool and eucalyptus is written in Perl and, due to the layout of the system, resides on the Bluegrit cluster. When the user presses the 'Compose Services' button on the tool, a remote connection is established with the Bluegrit cluster and the Perl code that parses the SLA RDF file is executed to find the suitable Eucalyptus configuration. To determine a suitable node the Perl code compares the number of cores

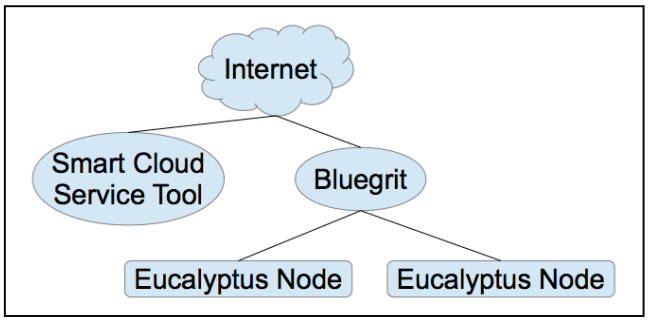

Figure 6: Interaction hierarchy between Smart Cloud Services tool and Eucalyptus cloud platform

and disk requirements specified in the SLA file to the ones provided by Eucalyptus. A match is considered successful if the number of cores and RAM provided by the Eucalyptus configuration is greater or equal to the request provided by the SLA and at least one instance of such a configuration can be started. If all the criteria are met, the software launches an instance of the configuration and reports back the IP address of the newly launched instance to the tool. The interaction with the Eucalyptus system is done via the command line tools provided by Eucalyptus. We used the command 'euca-describe-availability-zones' to determine what configurations were being provided by Eucalyptus and we used the 'euca-run-instances' to start up an instance once a matching configuration had been found. The parser software waits for the instance to start up by polling Eucalyptus every few seconds via the 'euca-describe-instances' command.

We discovered several shortcomings while interfacing with the Eucalyptus cloud platform. First, Eucalyptus only allows for five different configurations for Cloud instances. This limit makes it very hard for a user to request just the right configuration for their needs. A more flexible way of specifying exactly what a user's requirements are would be far more convenient. Besides the five different configurations provided by Eucalyptus and configured by the Eucalyptus administrator, it is impossible to configure a Eucalyptus node to meet the requested requirements. Hence our Perl code matches the SLA requirements to one of the five existing configurations provided by the Eucalyptus administrator. A configuration is considered a match if all the SLA requirements are met or exceeded. The configuration with the lowest requirements that still matches the SLA is chosen as the node. Second, the only configuration options provided by Eucalyptus were to vary the number of cores, the amount of RAM, and the size of local storage. In some circumstances it might be desirable to have additional limitations, such as requiring a specific CPU clock cycle or the availability of special hardware, such as GPUs. With the current Eucalyptus system such a request is not possible.

This paper describes our results with a single service provider Eucalyptus. In ongoing work more cloud providers (like IBM's Virtual Computing Lab VCL [27]) are being added to Bluegrit and this experiment will be repeated to

expand the ability of cloud users to provision cloud services composed using components provided by multiple service providers.

V. Conclusion and Future Work

In this paper we have described an integrated methodology to automate IT services lifecycle on the cloud. To the best of our knowledge, this is the first such effort, and it is critical as it provides a holistic view of steps involved in deploying IT services. Our methodology, described in section III, complements previous work on ontologies for service descriptions in that it is focused on automating the processes needed to procure services on the cloud. The methodology can be referenced by organizations to determine what key deliverables they can expect at any stage of the process. We also hope that it will enable the academia and the industry to be on the "same page" when referring to IT services on the cloud. The Smart Cloud Services tool successfully demonstrated how our methodology can be used to sig-nificantly automate the acquisition and consumption of cloud based services thereby reducing the large time required by companies to discover and procure cloud based services. We are in the process of releasing this tool to multiple users to analyze how this scales up. As part of our ongoing work in this area, we are working on integrating this tool with other cloud computing platforms available in the industry today. One of the first platforms that we are working on is the Virtual Computing Lab (VCL) [27] platform provided by IBM.

Acknowledgment

The authors would like to acknowledge Ms. Dawn Leaf, Cloud Computing Executive Program Manager at NIST, for her support and guidance in developing the Smart Cloud Services demonstration prototype.

References

[1] D. Bianchini, V. De Antonellis, B. Pernici, P. Plebani, Ontology-based methodology for e-service discovery, International Journal of Information Systems, The Semantic Web and Web Services, v. 31, n. 4-5, June-July 2006, pp 361-380

[2] J. Black et al, An integration model for organizing IT service Management, IBM Systems Journal, VOL 46, NO 3, 2007

[3] M. Hepp , "GoodRelations: An Ontology for Describing Products and Services Offers on the Web", Proceedings of the 16th International Conference on Knowledge Engineering and Knowledge Management (EKAW2008), Italy, 2008, Springer LNCS, Vol 5268, pp. 332-347.

[4] Jena– A Semantic Web Framework for Java, http://incubator.apache.org/jena/, retrieved on Feb 22, 2012.

[5] Joseki - A SPARQL Server for Jena, http://www.joseki.org/, retrieved on Feb 22, 2012.

[6] K. Joshi , T. Finin , Y. Yesha, "Integrated Lifecycle of IT Services in a Cloud Environment", in Proceedings of The Third International Conference on the Virtual Computing Initiative (ICVCI 2009), Research Triangle Park, NC, October 2009

[7] K. Joshi, OWL Ontology for Lifecycle of IT Services on the Cloud, 2010, http://ebiquity.umbc.edu/ontologies/itso/1.0/itso.owl

[8] K. Joshi, A. Joshi and Y. Yesha , "Managing the Quality of Virtualized Services", in proceedings of the SRII Global conference, San Jose, March 2011.

[9] O. Lassila, R. Swick and others, Resource Description Framework (RDF) Model and Syntax Specification, World Wide Web Consortium, 1999.

[10] D. Martin, et al., Bringing semantics to web services: The OWL-S approach, Lecture Notes in Computer Science, volume 3387, pp. 26-42, 2005, Springer.

[11] E. M. Maximilien, M.Singh, A Framework and Ontology for Dynamic Web Services Se-lection, IEEE Internet Computing, vol. 8, no. 5, pp. 84-93, Sep./Oct. 2004

[12] D. McGuinness, F. Van Harmelen, et al., OWL web ontology language overview, W3C recommendation, World Wide Web Consortium, 2004.

[13] NIST Special Publication 800-145, "The NIST Definition of Cloud Computing", Sep 2011.

[14] NIST Special Publication 500-292, "NIST Cloud Computing Reference Architecture", Nov 2011.

[15] Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L., Zagorodnov, D., "The eucalyptus open-source cloud-computing system", 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, pp 124-131, 2009

[16] M. Papazoglou and W. Van Den Heuvel, Service-oriented design and development methodology, International Journal of Web Engineering and Technology, Volume 2, Number 4, 2006, pp. 412 – 442

[17] S. Paurobally, V. Tamma and M. Wooldrdige, A Framework for Web Service Negotiation, ACM Transactions on Autonomous and Adaptive Systems,Vol. 2, No. 4, Article 14, November 2007.

[18] E. Prud'hommeaux and A. Seaborne, SPARQL Query Language for RDF, , W3C recommendation Jan 2008, http://www.w3.org/TR/rdf-sparql-query/, retrieved on April 27, 2011

[19] S Ran, A model for web services discovery with QoS, ACM SIGecom Exchanges, Vol 4, Issue 1, 2003, pp 1-10, 2003

[20] M. L. Sbodio, D. Martin, and C. Moulin, "Discovering Semantic Web services using SPARQL and intelligent agents." Journal of Web Semant. 8, 4 (November 2010), 310-328.

[21] R. Smith, The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver, IEEE Transactions on computers, Volume C-29, Issue 12, 1980, pp 1104-1113.

[22] SPARQL Endpoint, http://semanticweb.org/wiki/SPARQL_endpoint, retrieved on Feb 22, 2012.

[23] Van Bon et al., Foundations of IT service management based on ITIL V3 , Van Haten Publishing, 2008

[24] J. Cardoso, A. Barros, N. May, and U. Kylau, Towards a Unified Service Description Language for the Internet of Services: Requirements and First Developments, IEEE Intl Conference on Services Computing, IEEE Computer Society Press, 2010.

[25] Paliwal, A.; Shafiq, B.; Vaidya, J.; Xiong, H.; Adam, N.; , "Semantics Based Automated Service Discovery," Services Computing, IEEE Transactions on , vol.PP, no.99, pp.1, 0, doi: 10.1109/TSC.2011.19

[26] NIST Cloud Computing Use Case 3.9: Query Cloud-Provider Capabilities and Capacities, http://www.nist.gov/itl/cloud/3_9.cfm, retrieved on February 25, 2012

[27] IBM VCL : A Cloud Computing Solution in Universities, http://www.ibm.com/developerworks/webservices/library/ws-vcl/, retrieved on February 25, 2012

[28] L Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, Q. Sheng, Quality driven web services composition, Proceedings of the 12th international conference on World Wide Web, 2003, pp 411 - 421 .

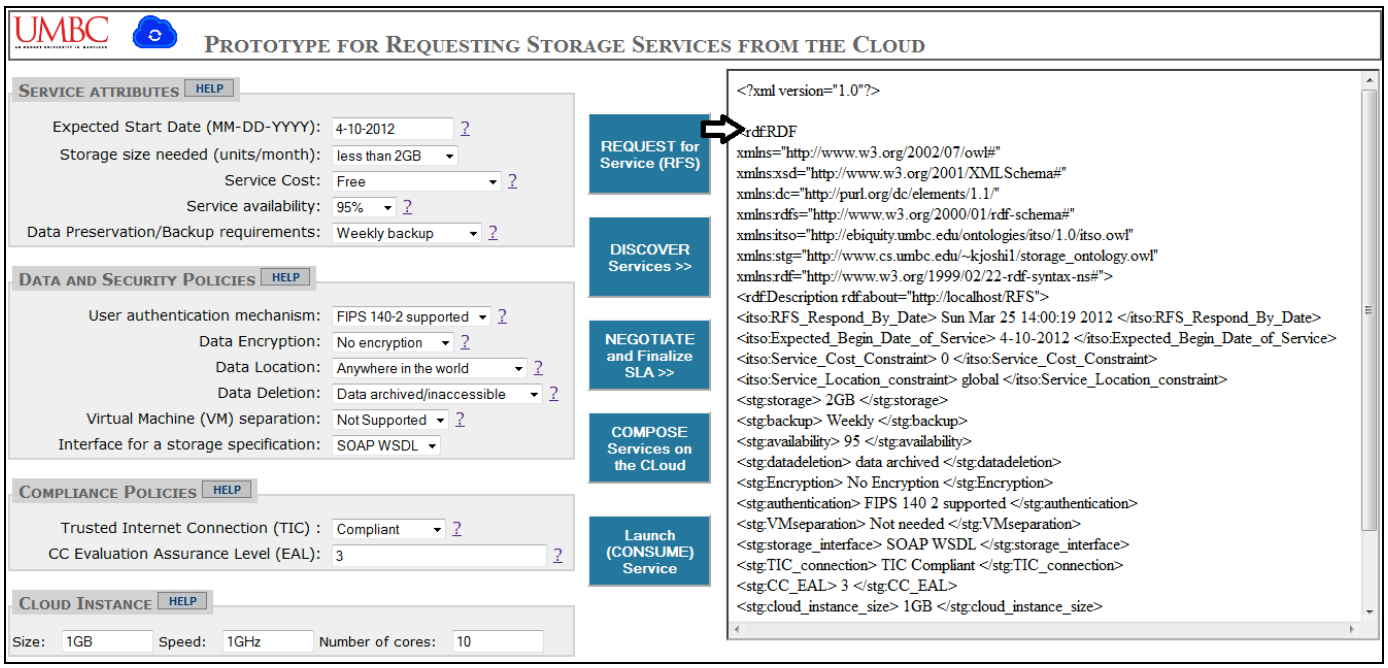

Figure 2: User Interface for Smart Cloud Service allows users to specify the service constraints using drop down lists and generate a Request for Service.

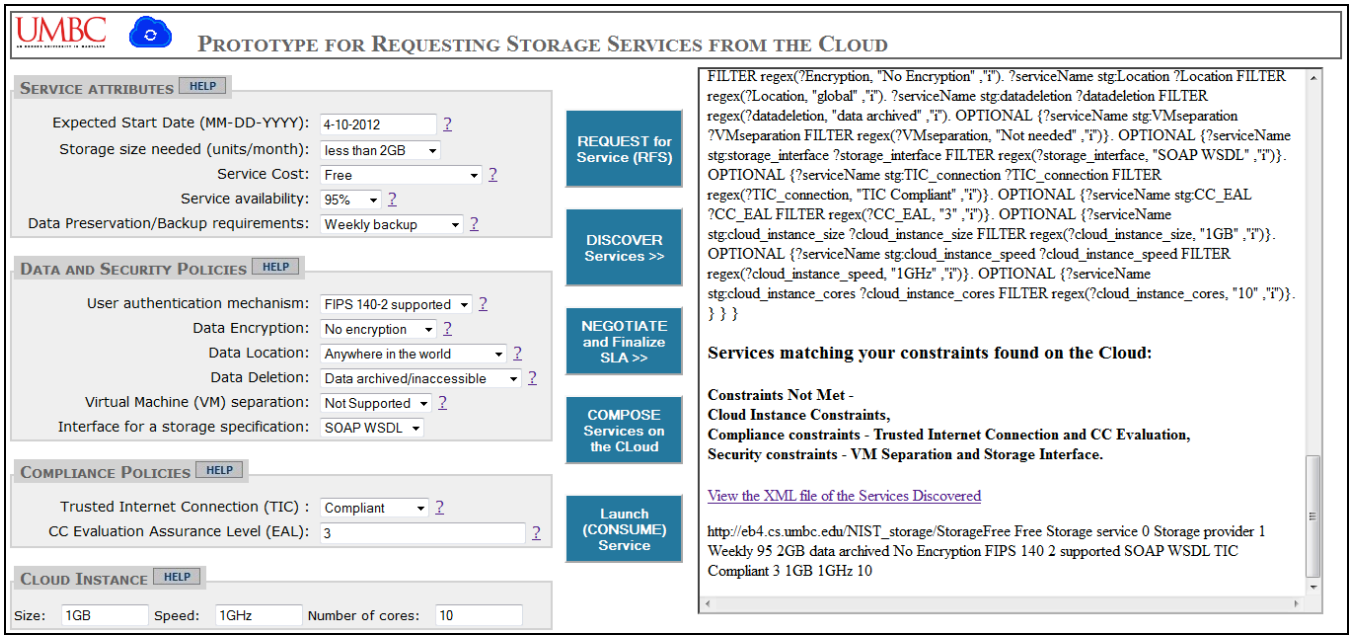

Figure 3: The tool relaxes RFS constraints iteratively during Service Negotiation

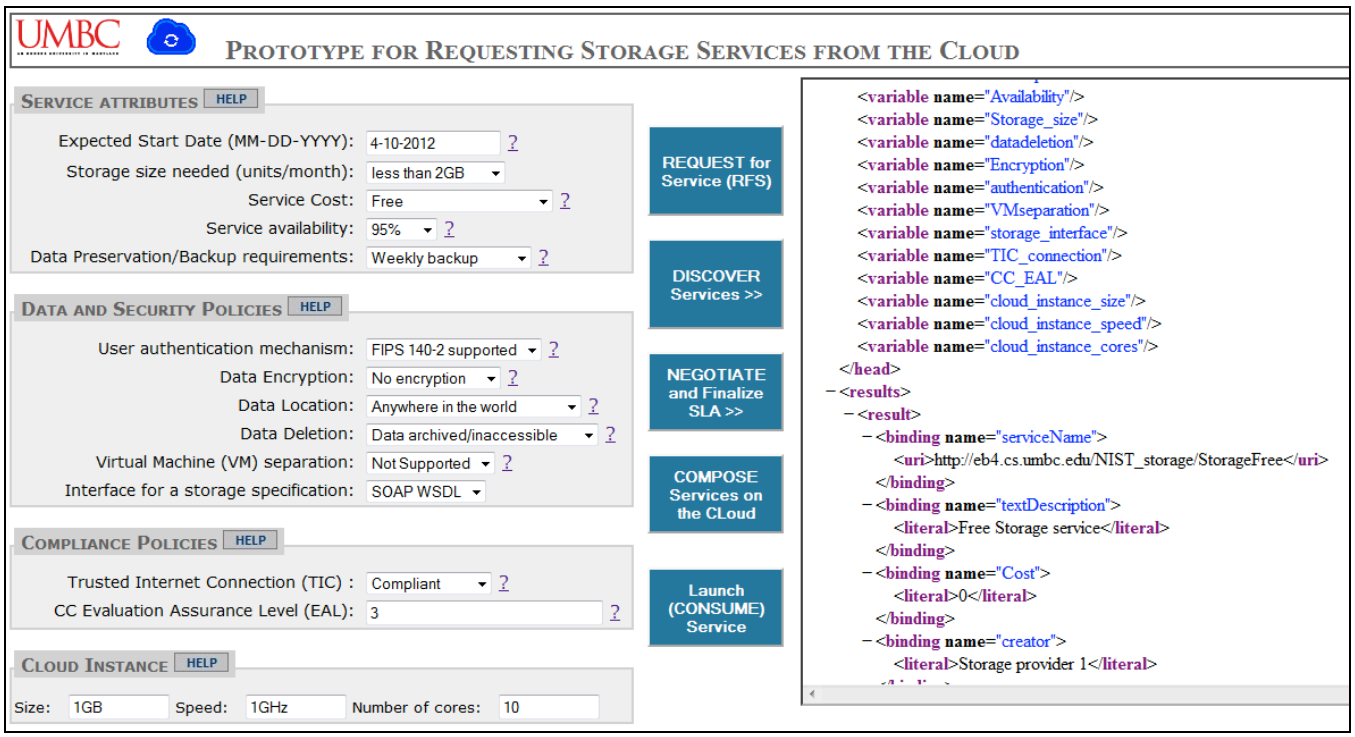

Figure 4: Matching Services are returned along with service IRI and constraint value.