Technical Report TR-CS-11-02

A Semantic Approach to Automate Service Management in the Cloud

Karuna P Joshi, Tim Finin, Yelena Yesha
Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County
Baltimore, MD 21250, USA

1 June 2011

COLLEGE OF ENGINEERING

UMBC

A Semantic Approach to Automate Service Management in the Cloud

Karuna P Joshi, Tim Finin and Yelena Yesha
Computer Science and Electrical Engineering Department
University of Maryland, Baltimore County
Baltimore, MD, USA
{kjoshi1, finin, yeyesha} @umbc.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. In this paper, we describe an integrated methodology for the lifecycle of IT services delivered on the cloud. We have divided the IT service lifecycle into five phases of requirements, discovery, negotiation, composition, and consumption. We detail each phase and list the high level ontologies that we have developed for them. We also describe a prototype system that we have developed using Semantic Web technologies to represent and reason about services and service requirements. This methodology complements previous work on ontologies for service descriptions in that it is focused on supporting negotiation for the particulars of a service and going beyond simple matchmaking.**

Keywords- Services; Methodology; lifecycle; Semantic Web; Ontologies.

I. INTRODUCTION

The development and maintenance of Information Technology (IT) was in the past regarded as an integral part of any organization. It has now been outsourced by most companies to external consulting/staffing companies or service providers. This outsourcing model initially led to off-shoring of tasks to providers due to their specialized expertise and/or labor arbitrage. It is now being replaced by a new delivery model where businesses purchase IT components like software, hardware or network bandwidth as services from providers, who can be based anywhere in the world. The service is acquired on an as needed basis and can be termed as *service on demand*. Typically the service is delivered to the organization via the Internet or mobile devices.

In such scenarios, 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 other cases, 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 other services and assets (technological, human, or process) that may be supplied by one or more service providers distributed across the network – in the cloud. Moreover, a single service could be a part of many composite services as needed. The service, in effect, is virtualized on the cloud. This virtualized model of service delivery potentially allows for easier service customization, better resource utilization and greater responsiveness on part of the service providers. It 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 managing services on the cloud is the lack of an integrated methodology for service creation and deployment that would provide a holistic view of the service lifecycle on a cloud. In this paper we present a methodology to address the lifecycle issue for virtualized services delivered from the cloud. We use semantically rich descriptions of the requirements, constraints, and capabilities that are needed by each phase of the lifecycle. This methodology is complementary to previous work on ontologies, like OWL-S, for service descriptions in that it is focused on automating processes needed to procure services on the cloud. We concentrate on enabling multiple iterations of service negotiation with constraints being relaxed with every iteration till a service match is obtained. In section III, we present the high level ontologies that we have created for the various phases in this paper, and show where existing ontologies can be leveraged. These can be reasoned over to automate the phases guided by high level policy constraints provided by consumers, service customers, or service providers. The proposed methodology will enable practitioners to plan, create and deploy virtualized services successfully. We have used Semantic Web technologies like OWL, RDF, and SPARQL to develop the prototype for our lifecycle prototype system described in detail in section IV of this paper.

II. RELATED WORK

At present there is no integrated methodology for the entire service lifecycle covering service planning, development and deployment in virtualized environments. Most approaches are limited to exploring a single aspect of the lifecycle like service discovery, service composition or service quality. 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.

Authors would like to acknowledge MURI grant 00003857 from Air Force Office of Scientific Research for this research.

Papazoglou and Heuvel [1] have proposed a methodology for developing and deploying web services using service oriented architectures. 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. [2], do not provide this flexibility and are limited to cases where a single service provider provides one service. Zeng et al. [3] 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 [4] 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. [5] 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. [14] have described a framework for negotiation of web services using the iterated Contract Net Protocol (CNP). 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 [24] 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.

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 [7] 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.

In a virtualized service-oriented environment, consumers and providers need to be able to exchange information, queries, and requests with some assurance that they share a common meaning. This is critical not only for the data but also for the policies followed by service consumers or providers. One possible approach to this issue is to employ Semantic Web techniques for modeling and reasoning about services related information. We have used this approach for our services lifecycle and prototype development.

The Semantic Web is an enhancement of the World Wide Web that 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) [11] and Web Ontology Language (OWL) [9] for defining ontologies and describing meta-data using these ontologies as well as tools for reasoning over these descriptions. Semantic Web knowledge can also be encoded in rule format using several approaches, including N3-logic [16] and SWRL [17]. The Ontology Web Language for Services (OWL-S) [18] 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 describing the functions they provide. These 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.

III. PROPOSED ONTOLOGY

We divide the IT service lifecycle on a cloud into five phases as depicted in Figure 1. In sequential order of execution they are requirements, discovery, negotiation, composition, and consumption. We have described these phases in detail along with the associated metrics in [25]. In the following sections we present the pictorial representations of high-level ontologies that we have created for each phase. We have developed the ontology for the entire lifecycle in OWL 2 DL profile and it can be accessed at [27].

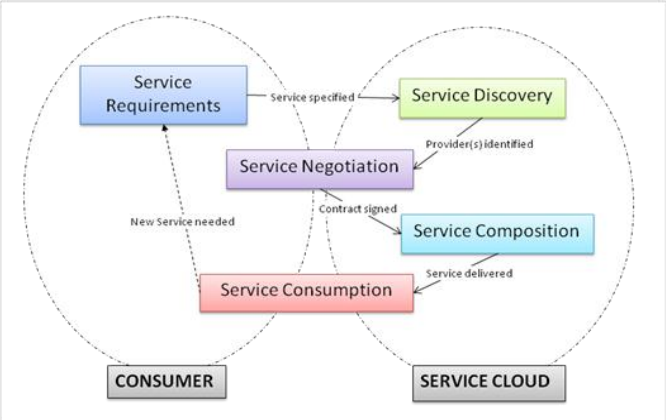

Figure 1: The IT service lifecycle on a virtualized cloud comprises five phases: requirements, discovery, negotiation, composition and consumption

A. *Service Requirements Phase*

In this phase the consumer details the technical and functional specifications that a service needs to fulfill. While defining the service requirements, the consumer specifies not just the functionality, but also non-functional attributes such as characteristics of the providing agent, constraints and preferences on data quality and required security policies for the service. Service compliance details like required certifications, standards to be adhered to etc. are also identified. Depending on the service cost and availability, a consumer may prefer to procure service

with lower quality data. This benefits the service provider as well since low quality data requires low maintenance. 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.

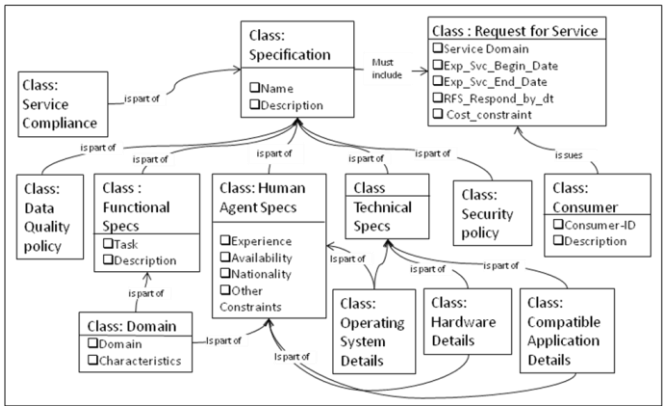

Figure 2: Ontology of service requirements phase contains the RFS class which includes an instance of the Specification class

Figure 2 illustrates the high level ontology for this phase. The two top classes are the Specification class and the "Request For Service" class. The Specification class consists of six main classes that define the functional specifications, technical specifications, Human agent specifications, security policies, service compliance policies and data quality policies. The technical specs contain information about the Hardware, Operating System and other compatible services/applications that the desired service should conform to. Human Agent specifications also list the technical and domain expertise that the service providing agent should have. Part of our ongoing work is to use existing ontologies that have been developed for classes like standard hardware, operating systems and computer applications.

```
// allows services of low data quality to be accepted if the service is
free of cost.
@forAll  :SERVICE_QUALITY, :COST.
:data_quality_policy a air:Policy;
          rdfs:label "Data Quality policy";
          air:rule :data-quality-rule.
:data-quality-rule a air:Belief-rule;
        rdfs:label "data quality rule";
        air:pattern {
          :SERVICE_QUALITY quality_level :LOW.
          :COST service_cost :0.
        };
air:assert {:SERVICE_QUALITY air:compliant-
with  :data_quality_policy.};
air:alt { air:assert {:SERVICE_QUALITY air:NOTcompliant-
with :data_quality_policy.}}.
```

Figure 3: Describing Service specifications and constraints using AIR Policy Language

Semantic Web policy languages can be used to describe service specifications and constraints in machine-processable format. For instance, a consumer may opt for a service with poor data quality to take advantage of low cost of the service. This policy can be described by using the AIR [22] policy language as shown in Figure 3.

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 search engines or service brokers can be employed. This engine runs a query against the services registered with a central registry or governing body and matches the domain, data type, compliance needs, functional and technical specifications and returns the result with the service providers matching the maximum number of requirements listed at the top.

One critical part of this phase is service certification, in which the consumers will contact a central registry, like UDDI [6], to get references for providers that they narrow down to.

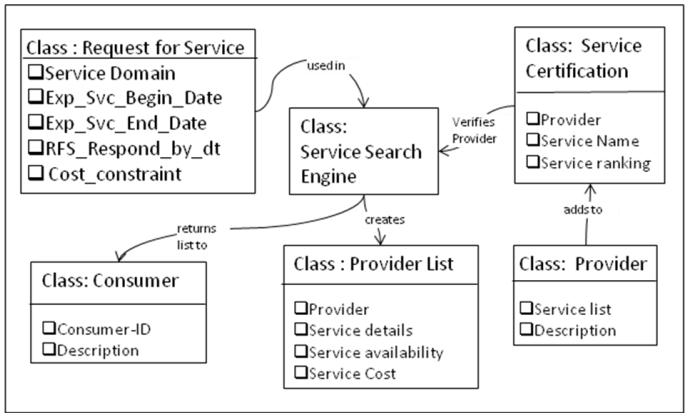

Figure 4: Ontology for service discovery phase uses the RFS class to search for providers and generate a Provider list

Figure 4 illustrates the high level ontology for the service discovery phase, which uses the RFS class from the requirements phase to search for service providers and generate a list of providers with which to begin negotiations. The class Service certification validates the provider's credentials.

If the consumers find the exact service within their budgets, they can begin consuming the service immediately upon payment. However, often the consumers 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 begun negotiations with the service providers which is the next phase of the lifecycle. 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) [10] 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, 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 negotiation steps are listed below and shown in the negotiation sequence diagram in Figure 5.

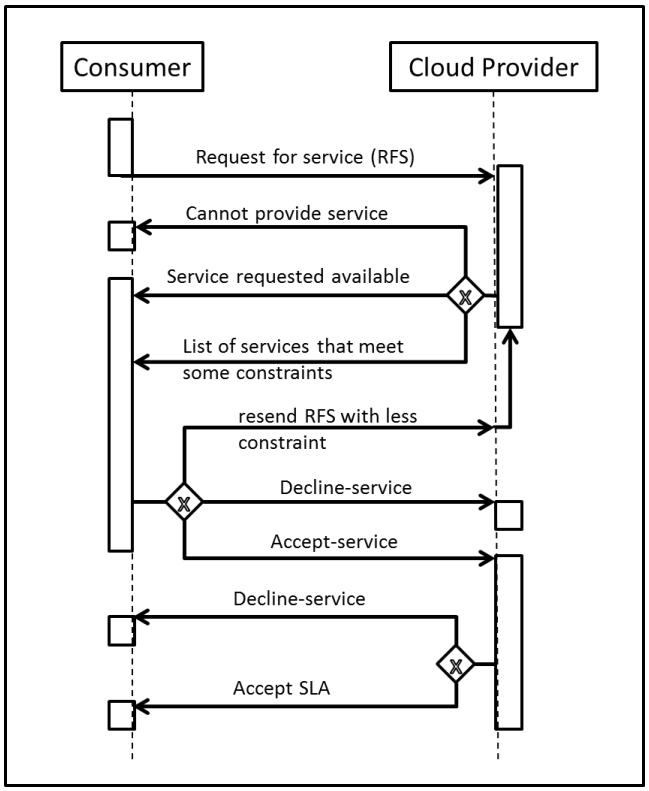

Figure 5: Sequence diagram showing the steps of the service Negotiation

1. The consumer sends a RFS to the provider specifying the functional and non-functional requirements.

2. The provider responds to the RFS in one of three ways

2a) Informs the consumer that it cannot provide the service, terminating negotiation.

2b) Indicates that a service matching all the requirements and constraints exists and sends the quote with SLAs.

2c) Indicates that there is a partial match of requirements and sends the quote with SLA file listing matching constraints.

3. The consumer receives and considers the quote

4. The consumer responds to the quote in one of three ways

4a) If the quote is a partial match, the consumer relaxes the service constraints and/or functionality and resends the RFS to the provider. The provider repeats the actions in step 2.

4b) If the response is a full match and the consumer is satisfied with the offer then negotiation is regarded complete. The consumer signs this offer and returns it as an SLA.

4c) The consumer can decline the service, terminating the negotiation.

5. The provider responds to the RFS in one of two ways

5a) The provider can no longer provide the service, and rejects the agreement, terminating negotiation.

5b) The provider agrees with the constraints, and the same RDF file consisting of the SLA now exists with both parties.

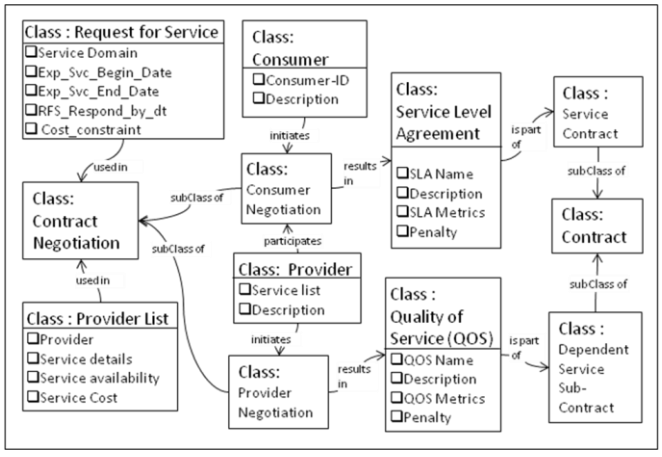

Figure 6: Ontology for service negotiation uses the RFS class for the contract negotiation and creation of SLA and QoS

We have constructed a high level ontology for this phase and it is illustrated in Figure 6. This phase uses the RFS class from the requirements phase and the provider's list class from the discovery phase to negotiate the contracts between consumer and primary provider and between the various component providers themselves. 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.

Figure 7 illustrates the high level ontology for this phase. The main class of this phase is the Service class that combines the various components into a single service. We include the OWL-S Composite Process class ontology. The Service class takes inputs from the Specification, Service Contracts and Service Level Agreement classes defined in the earlier phases to help determine the orchestration of the various components.

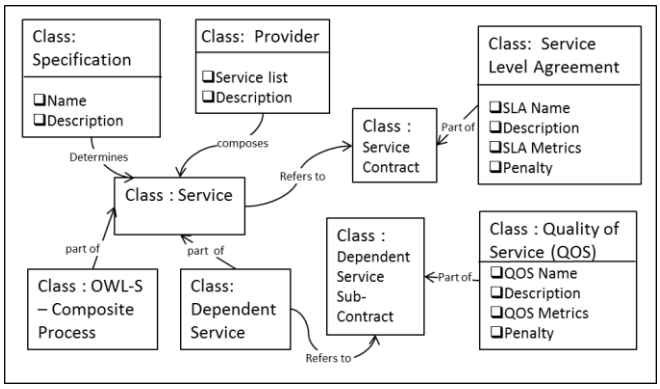

Figure 7: Ontology for composition phase builds on the OWL-S composite process class.

Once the service is composed, the lifecycle enters the final phase of service consumption which is detailed in the next section.

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 for the same. The consumer then begins consuming the service. In a cloud environment, the service usually resides on remote machines managed by the service providers. Hence the onus for administrating, managing and monitoring the service lies with the provider. 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.

Figure 8 illustrates the ontology for this phase. The composite service is composed 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 [26].

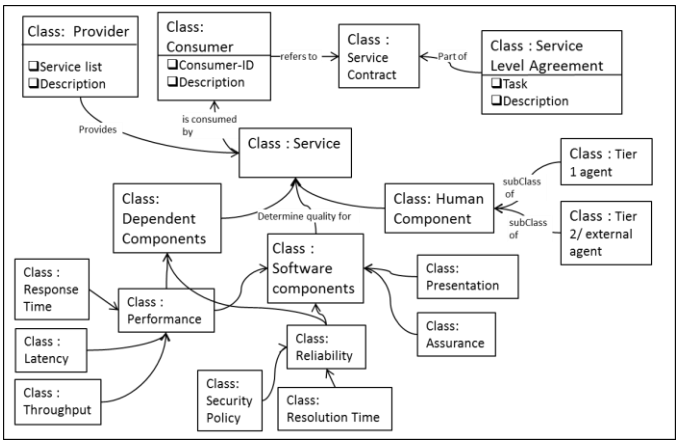

Figure 8: Ontology for consumption phase contains classes to monitor the quality of software, human and dependent components of the composite process

IV. PROTOTYPE

In this section we describe the prototype that we have constructed as a proof of concept for our proposed lifecycle. We have selected a simple storage service for our prototype. We have built the prototype by using SPARQL Protocol and RDF Query Language (SPARQL) [12] end points to simulate service providers and consumers spread across the cloud. We are using Jena Semantic Web framework [21] and the Joseki software [13], which is a HTTP engine that supports the SPARQL Protocol and the SPARQL RDF Query language, to develop our prototype.

A. SPARQL and SPARQL Endpoints

SPARQL is the query language for RDF that has been standardized by W3C [12]. 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 SPARQL queries can be results sets or RDF graphs. A SPARQL abstract query is defined in [12] 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 [19][20]. 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. Service Descriptions [20] specify the capabilities of a SPARQL endpoint. They provide a declarative description of the data available from an endpoint, the definition of limitations on access patterns and statistical information about the available data that is used for query optimization.

B. Example: Storage Service

For our illustration, we consider a basic storage service where consumers can store their files or applications on the cloud. This service is currently very popular on the cloud and is offered by multiple providers, each offering different storage and pricing plans.

We will use the service lifecycle ontology that we described in the previous section to acquire the service. To describe the technical specifications for the RFS in the requirements phase, we however need to create a domain specific ontology defining the key properties of the service like storage size, backup etc. For this illustration, we have created a basic ontology to define the technical specifications of the storage service and it is illustrated in figure 9. The service consists of three attributes, viz. storage size which indicates the storage space in Gigabytes that consumer wishes to procure; the Cost per Unit and the Backup attribute which indicates whether the files/applications stored using the service are backed up or not. Storage services offered currently by companies like Amazon etc. have many more attributes associated with the service, like data I/O, redundant storage etc. For simplicity sake and to illustrate the iterative negotiation process, we are limiting our service to three attributes. We will be using our service lifecycle ontology and the OWL-S ontology in conjunction with this ontology for our prototype.

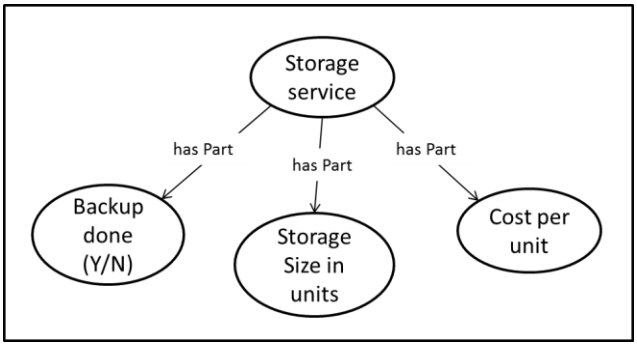

Figure 9: Ontology defining the technical specifications for a simple Storage service

After defining our service, we created a SPARQL endpoint using Joseki to simulate a service provider providing the storage 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 contains the service description along with the provider policies endpoint. Using the N3 [16] notation, the storage service description is described in figure 10.

```
@prefix owls:   <http://www.ai.sri.com/daml/services/owl-s/1.2/Profile.owl>.
@prefix sv:   <http://www.cs.umbc.edu/~kjoshi1/IT_Service_Ontology.owl>.
@prefix storage:  <http://www.cs.umbc.edu/~kjoshi1/storage_ontology.owl>.
@prefix :          <http://eb4.cs.umbc.edu/Storage/>.

# A database of storage services and their providers
:StorageFree
   owls:serviceName    "Storage 1" ;
   owls:textDescription    "Basic  Storage service – free of cost " ;
   sv:creator  "provider1" ;
   storage:Cost  "0" ;
   storage:Backup  "No" ;
   storage:Storage_size  "< 5GB" ;
.

:Storage10GB
   owls:serviceName    "10 GB Storage" ;
   owls:textDescription    "Storage up to 10 GB" ;
   sv:creator  "provider1" ;
   storage:Cost  "$0.14 per GB per month" ;
   storage:Backup  "Yes" ;
   storage:Storage_size  "< 10 GB" ;
    .

:Storage100GB
   owls:serviceName    "100GB Storage" ;
   owls:textDescription    "S3 Storage service for less than 49 TB" ;
   sv:creator  "provider1" ;
   storage:Cost  "$0.125 per GB per month" ;
   storage:Backup  "Yes" ;
   storage:Storage_size  "<100 GB" ;
```

Figure 10: Service description of the Storage service in N3 notation on a SPARQL endpoint

C. Service Discovery

We created multiple SPARQL endpoints to simulate both service consumers and providers. We used federated SPARQL queries, like those illustrated in Figure 11, to discover the services residing on an end point . Researchers like Sbodio et. al [23] have also proposed algorithms for service discovery using SPARQL language.

```
PREFIX owls: <http://www.ai.sri.com/daml/services/owl-s/1.2/Profile.owl>
PREFIX sv:   <http://www.cs.umbc.edu/~kjoshi1/IT_Service_Ontology.owl>
PREFIX stg:  <http://www.cs.umbc.edu/~kjoshi1/storage_ontology.owl>
SELECT  ?serviceName ?textDescription ?Cost ?creator ?Backup ?Stor-
age_size
 { SERVICE <http://eb4.cs.umbc.edu:2020/Storage>
   { SELECT  ?serviceName ?textDescription ?creator ?Cost ?Backup  ?Stor-
age_size
    WHERE
    {?serviceName owls:textDescription ?textDescription .
     ?serviceName stg:Cost ?Cost .
     ?serviceName stg:Backup ?Backup .
     ?serviceName sv:creator ?creator .
     ?serviceName stg:Storage_size ?Storage_size .
    }
   }
  }
```

Figure 11: Service Discovery by using SPARQL query to get service description

D. Service Negotiation

In our example, the consumer will negotiate directly with a cloud provider to acquire the service and determine the SLAs No third-party or cloud brokers are involved in the negotiation. The contract is defined through an SLA between the provider and the end-user.

In the requirements phase, the consumer identifies all the constraints or assertions of a service that need to be met along with its functional requirements. These constraints typically can be classified as hard and soft constraints. Hard constraints are non-negotiable and have to be met by service providers. Soft constraints help to define the desired service attributes on which the consumer is willing to negotiate. Often, the same parameter will have both a hard and a soft constraint (e.g., a desired software version vs. the minimum version needed, a desired amount of memory vs. the minimum required, etc.). A policy driven approach can both capture such constraints, and also guide the negotiation between the consumers and the providers. For our example, the negotiation policies would be:

HARD REQUIREMENTS:
- MINIMUM STORAGE NEEDED: 100 GB
- COST: AT MOST $8/Month
- BACKUP: YES

SOFT REQUIREMENTS:
- STORAGE NEEDED: 200 GB
- COST: $5/MONTH
- BACKUP: YES

The consumer's requirements policy (in the RFS) specifies the soft requirements. When the provider's service policy manager reviews the requirements and finds that it can't meet them, it will try and negotiate. The consumer's response to the counterproposals meeting its hard requirements would be guided by the policy which ranks the constraints. For instance, the policy might ask it accept disk storage as close to the minimum needed as possible to keep the cost low. Figure 12 illustrates a RDF graph of the SLAs that will be finalized at the end of the negotiation phase for the constraints listed above.

The SLA graphs along with the functional and technical specifications will determine the service components and their orchestration in the service composition phase. The SLAs will also contain the metrics that will be used to manage service performance during the final service consumption/monitoring phase. In our illustration, the service performance will be regarded poor if either of the attributes - size, cost and backup - falls below the specified levels.

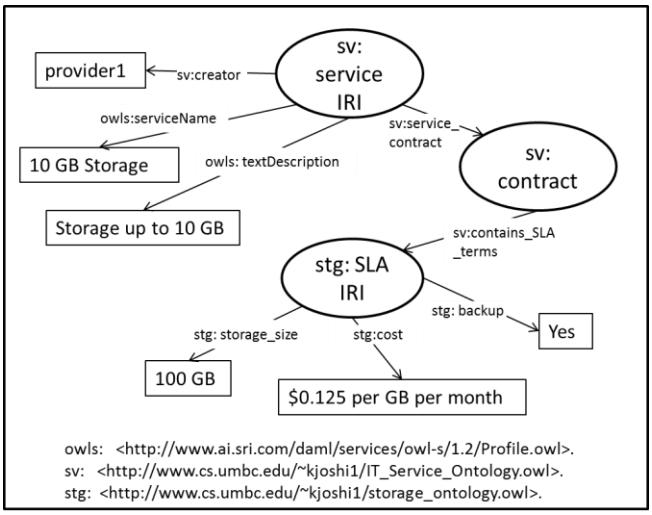

Figure 12: RDF graph representing the SLAs agreed upon for our example during the service negotiation phase

V. Conclusion and Ongoing Work

In this paper we have defined an integrated ontology for processes needed 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 approach 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 they speak about IT services on the cloud. We are currently refining the ontology to capture key metrics of software quality, as well as their relations. We are also identifying the SAWSDL [8] extensions for each phase. We are also working on automating the negotiation steps in service acquisition and refining our prototype to

demonstrate the various phases and plan on using actual Enterprise policies to demonstrate the validity of this framework.

REFERENCES

[1] 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

[2] 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, Volume 31, Issues 4-5, June-July 2006, pp 361-380

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

[4] 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

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

[6] S Ran, A model for web_services discovery_with QoS, ACM SIGecom Exchanges, Vol 4, Issue 1, 2003, pp 1-10, 2003

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

[8] J. Kopecky, T. Vitvar, C. Bournez and J. Farrell, SAWSDL: Semantic annotations for WSDL and XML schema, IEEE Internet Computing, v11n6, pp. 60-67, 2007.

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

[10] 'Whats in a Service Level Agreement?', SLA@SOI, http://sla-at-soi.eu/?p=356, retrieved on July 30, 2009.

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

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

[13] Joseki - A SPARQL Server for Jena, http://www.joseki.org/, retrieved on April 27, 2011.

[14] 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.

[15] 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.

[16] T. Berners-Lee, D. Connolly, L. Kagal, Y. Scharf and J. Hendler, N3Logic: A logical framework for the World Wide Web, Theory and Practice of Logic Programming, v8n3, pp. 249-269, Cambridge Univ Press, 2008.

[17] I. Horrocks, P. Patel-Schneider, H. Boley, S. Tabet, B. Grosof and M. Dean, SWRL: A semantic web rule language combining OWL and RuleML, W3C Member Submission, World Wide Web Consortium, 2004.

[18] 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.

[19] SPARQL Endpoint, http://semanticweb.org/wiki/SPARQL_endpoint, retrieved on May 10, 2011.

[20] G. Williams, SPARQL Service Description, http://www.w3.org/TR/-sparql11-service-description/, retrieved on May 10, 2011.

[21] Jena– A Semantic Web Framework for Java, http://jena.sourceforge.net/, retrieved on May 10, 2011.

[22] L. Kagal, C. Hanson, and D. Weitzner, "Using dependency tracking to provide explanations for policy management", IEEE International Workshop on Policies for Distributed Systems and Networks, 2008.

[23] 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.

[24] 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.

[25] K. Joshi, T. Finin, and 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

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

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