Automating Cloud Services Life Cycle through Semantic Technologies

Karuna P. Joshi, Yelena Yesha, and Tim Finin

Abstract—Managing virtualized services efficiently over the cloud is an open challenge. Traditional models of software development are not appropriate for the cloud computing domain, where software (and other) services are acquired on demand. In this paper, we describe a new integrated methodology for the life cycle of IT services delivered on the cloud and demonstrate how it can be used to represent and reason about services and service requirements and so automate service acquisition and consumption from the cloud. We have divided the IT service life cycle into five phases of requirements, discovery, negotiation, composition, and consumption. We detail each phase and describe the ontologies that we have developed to represent the concepts and relationships for each phase. To show how this life cycle can automate the usage of cloud services, we describe a cloud storage prototype that we have developed. 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.

Index Terms—Intelligent web services and Semantic Web, life cycle, ontology design, web-based services

✦

1 INTRODUCTION

REGARDING information technology (IT) as a service delivered to the end user is a paradigm shift that is fast changing the way businesses looks at the role of IT within the organization. The outsourcing model is 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 hosted on a cloud or a computing grid and 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 component could be a part of many composite services as needed. The service, in effect, is virtualized on the cloud [38]. It is becoming the preferred method to deliver services ranging from help desk 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.

One consequence of this development is that the consumers now have more choices of service providers that they can select from. However, at present, most of the services are delivered as web services providing a singular functionality. Often, the onus is on the consumer to procure these web services individually and then integrate them per the requirements. There has been some work in creating brokers that would perform this functionality. However, such brokers work only on a fixed, linear description of service functionality which often fails to capture the complete requirements of the service needed, and the flexibility a consumer might have. In order to be able to take advantage of virtualized service models, it is imperative for the consumer to be able to identify all the constraints or assertions of a service that need to be met along with its functional requirements.

In our discussions with large organizations interested in acquiring cloud services, especially from public cloud providers, we have observed that a key barrier preventing organizations from successfully managing virtualized 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 life cycle on a cloud. In this paper, we present a methodology to address the life-cycle 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 life cycle. This methodology is complementary to previous work on ontologies, like Ontology Web Language for Services (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 iteratively till a service match is obtained. In Section 3, we present the high-level ontologies that we have created for the various phases in this paper,

• The authors are with the Computer Science and Electrical Engineering Department, University of Maryland, Baltimore County, 1000 Hilltop Circle, Baltimore, MD 21250. E-mail: {kjoshi1, yeyesha, finin}@umbc.edu.

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.

The key reason to have a semantically rich approach to describe cloud attributes and service-level agreements (SLA) is to permit distributed clients and cloud service providers to "automate" the process of acquisition and consumption of services. Without a semantic approach that will permit the providers and consumers to understand each other, which is the present state of the practice, the acquisition process is done manually, and the consumption/monitoring process also requires significant manual input. For instance, the National Institute of Standards and Technology (NIST) has identified ambiguity in cloud SLAs currently offered by cloud providers as one of the factors that prevent broad cloud adoption by large organizations, especially federal agencies [46]. It is very difficult to compare SLAs offered by two cloud providers to determine who is offering the better deal. Also, existing cloud SLAs (for instance, SLA provided by Amazon at http://aws.amazon.com/ec2-sla/) are provided as a text document making it open to interpretation and very difficult to monitor SLA performance and adherence by the cloud provider. Additionally, survey of industry sources also indicates overall dissatisfaction among cloud users of existing cloud SLA.

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 attributes, security policies, and compliance policies via a simple user interface. We used W3C standard Semantic web technologies, such as Web Ontology Language (OWL) [18], Resource Description Framework (RDF) [15], and SPARQL [24], to develop our prototype system since they enable us to build the vocabulary (or ontology) of our service life cycle 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 [18] 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 that guarantee sound and complete reasoning with various levels of complexity (e.g., N2ExpTime for OWL 2 DL). Moreover, there are also profiles that are tuned to work well with popular implementation technologies, for example, 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 webpages. Microdata has been adopted by Schema.org, 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, and so on. Several database vendors, including Oracle and IBM, have sophisticated support for representing RDF and OWL, including reasoning.

2 RELATED WORK

Since cloud computing is a nascent field, there is lack of standardization and a need has been felt to clearly define its key elements. NIST has recently released a special publication 800-145 [19] 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. Currently, it is very difficult for organizations to specify their data, security, privacy, and compliance policies, while automatically provisioning cloud services. We have addressed this in our proposed framework described in the next section.

In addition to the standard definition of cloud computing, NIST has also released the cloud computing reference architecture [20] document 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. We have referenced the NIST cloud computing standards to develop our ontology that is described in the next section.

Current research on cloud or web services so far has been limited to exploring a single aspect of the life-cycle-like service discovery, service composition, or service quality. There is no integrated methodology for the entire service life cycle 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 [22] 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. [3], do not provide this flexibility and are limited to cases where a single service provider provides one service. Zeng et al. [33] 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 [17] 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. [4] 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. [23] have described a framework for negotiation of web services using the iterated Contract Net Protocol (CNP) [29]. However, their implementation is limited to pre-existing web services and does not 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. Good-Relations [6] 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. Ren et al. [34] have proposed a technique for more efficient composition of semantic services.

Research on grid computing has also examined issues on on-demand provisioning and service discovery/composition [39], [40], [41], [42], [43], [44]. This research has primarily concentrated on addressing issues from cloud provider's perspective. Given the origins of grid computing in the scientific computing domain, this makes perfect sense. However, many issues related to policies of the consumer and the service acquisition processes are ignored. We approach the issue instead of looking at it from a holistic viewpoint of both the consumer as well as the provider. The authors have also not accounted for virtualized services that will be created by combining preexisting components.

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 [31] 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.

2.1 Semantic Web

As explained in Section 1, we have used Semantic Web technologies to develop the services life cycle and prototype development. Semantic Web enables data to be annotated with machine understandable metadata, allowing the automation of their retrieval and their usage in correct contexts. Semantic Web technologies include languages such as RDF [15] and OWL [18] for defining ontologies and describing metadata using these ontologies as well as tools for reasoning over these descriptions. OWL is based on Description Logic (DL) [1] with a representation in RDF. OWL Semantic Web knowledge can also be encoded in rule format using several approaches, including N3-logic rules [2], SWRL rules [7] and RIF, the new W3C standard for Rule Interchange Formalism. 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.

Several OWL ontologies have been developed to describe services, including Ontology Web Language for Services [16] and Semantic Annotations for WSDL and XML Schema (SAWSDL) [14]. 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 and it is described in Section 3.4. SAWSDL defines mechanisms using which semantic annotations can be added to WSDL components. Sheth et al. [27] describe the METEOR-S project that resulted in the submission of WSDLS specification which was used as the input for SAWSDL.

SPARQL Protocol and RDF Query Language (SPARQL) is the query language for RDF that has been standardized by W3C [24]. 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. The results of SPARQL queries can be results sets or RDF graphs. A SPARQL endpoint is a conformant SPARQL protocol service as defined in the SPARQL Protocol for RDF (SPROT) specification [30]. It enables users 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 toward a knowledge base. Service descriptions [32] 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.

3 PROPOSED SERVICE LIFE-CYCLE ONTOLOGY

Traditional models of software development, like the waterfall method or the spiral method [5], consists of phases like planning, analysis, design, testing, and acceptance. These methodologies are found to be very time-consuming and require extensive human labor, both from the software application consumer as well as the provider. Cloud computing environment promises agility, elasticity, and quick turnaround time for provisioning resources and services. Virtualized services that provide "on-demand" service on the cloud are mainly built by combining

preexisting components that are developed by same or multiple providers. In this scenario, following traditional software life-cycle methodologies will significantly slow the service delivery time, thereby rendering the service provider uncompetitive in the cloud market. Hence, we believe a radically different methodology is needed for cloud-based services.

To develop this ontology, we had detailed discussions with various large organizations who are interested in acquiring cloud-based services. Among our chief collaborators were NIST, our university's division of IT, and a large international financial organization with global presence. Additionally, one of the authors of this paper has had extensive experience in managing large IT services and we were able to draw on that experience while developing this framework. While developing the ontology, we referred to NIST's cloud computing reference architecture [20] to identify the key stakeholders in the life cycle.

We divide the virtualized service life cycle on a cloud into five phases. In sequential order of execution, they are requirements, discovery, negotiation, composition, and consumption. Our focus for this framework is the life cycle for virtualized cloud services, where the services are composed of combining preexisting components. Hence, this life cycle does not include any requirements analysis or design phases. We assume that services, that are designed using a variety of existing approaches, will be described using our ontology—something that can be done post facto—and will be discoverable using standard (web) service-type mechanisms (e.g., UDDI, SLP). We also permit these services to be arbitrarily composed to create new services. We argue that this hews closely to the cloud model—a provider has a set of available services which can be made available as is, or in combination with other services. We do not claim that cloud providers only offer preexisting component-based services; however, a survey of current cloud based offerings has shown us that majority of the cloud services consist of preexisting components with minimal configuration capability and so we gather this is what the consumers are currently interested in from cloud providers.

Our ontology does not describe the service, but defines the data and processes needed to automate the acquisition and consumption of cloud services. The processes of acquiring such services are largely independent of the type of cloud service (IaaS, software as a service (SaaS), platform as a service (PaaS)), cloud deployment (private, public, hybrid), or service domain (computing services, healthcare, financial services, etc.). Our framework assumes that users will be defining the ontologies for functional and technical specifications for the service which will obviously vary for different domains. There is a significant body of work (e.g., SAWSDL, WSDL-S, WSMO, etc.) that provides ontologies to describe specific services in terms of their functional and technical specifications. Our framework makes it possible to integrate these functional and technical specifications with other enterprise-specific policies defined using the ontologies we provide (like privacy, security, compliance, human agent policies) in the requirements phase. And so only the functional description of the requirements phase ontology will have to be defined for each service. Our prototype described in next section is an example, where, for completeness, we have also described the service itself to show how our overall framework would work.

We have described the five phases in detail along with the associated metrics in [10]. Fig. 1 is a pictorial representation detailing the processes and data flow of the five phases. 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 life cycle in OWL 2 DL profile and it can be accessed at [11].

3.1 Service Requirements Phase

In the service requirements phase, the consumer details the technical and functional specifications that a service needs to fulfill. While defining the service requirements, the consumer also specifies nonfunctional attributes like characteristics of the human agent providing the service, constraints and preferences on data quality, and required security policies for the service. Service compliance details like certifications needed, standards to be adhered to, and so on, are also identified. 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 RFS can be generated in a machine readable format using Semantic Web technologies and we have illustrated this in the next section.

Majority of the users will not have static requirements and might not be able to initially articulate all their needs. Also, the requirements will continue to evolve as users acquire more and more cloud services. Hence, our framework captures a "snapshot" of the user requirements via the RFS and initiates the service discovery process to acquire services that match that snapshot. If the user is not satisfied with the services discovered, they can change their requirements (say, by increasing the cost constraint) and/or policies and restart the discovery phase with a new RFS. We also assume that the user requirements will change once the user begins consuming the services and so we show a link between the consumption and requirements phase (see Fig. 1) to indicate system triggers that could start a new cycle of service acquisition with a different requirements snapshot (new RFS).

Some of the policies and constraints that may be included in RFS are listed below. Additional policies/constraints that may be domain specific can be specified as needed:

1. Functional specifications list:

   a. Service tasks to be provided.
   b. Budgetary/cost policies and constraints.
   c. Service domain.
2. Technical specifications:

   a. Service's software applications.
   b. Software compatibility constraints.
   c. Hardware policy—e.g., mobile device, PC, server, multicore, and so on.
   d. Operating system policy—e.g., single OS support, multiple OS support.
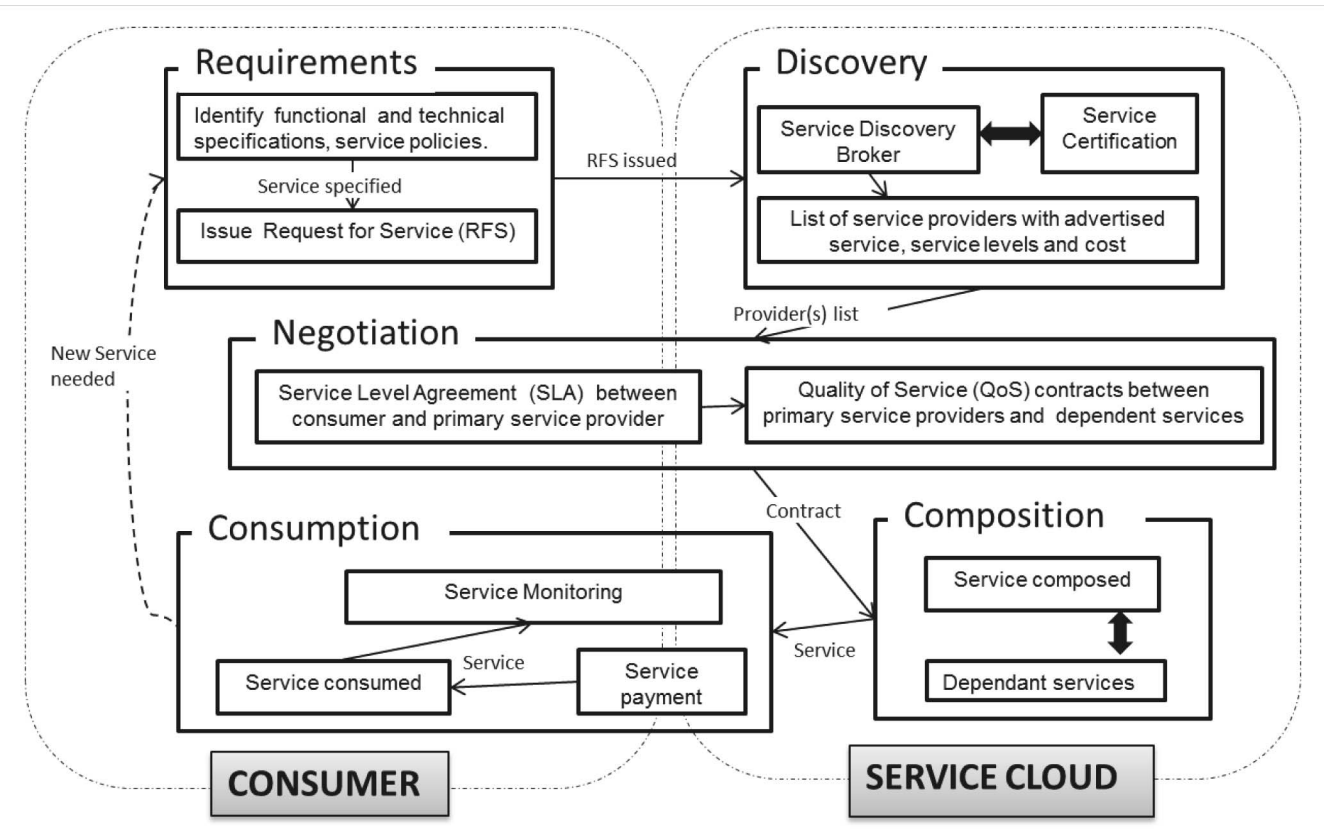   e. Language support policy.

Fig. 1. The IT service life cycle on a cloud comprises five phases: requirements, discovery, negotiation, composition, and consumption.

f. Cloud deployment—private, public, hybrid, community.
g. Cloud service layer—IaaS, PaaS, SaaS.
3. Human agent policy:

a. Agent experience in years.
b. Agent skill level.
c. Agent's location constraints.
d. Nationality/work permit constraints.
4. Security policy:

a. Roles and permissions
b. Cloud/service provider location constraints.
c. Data encryption, deletion constraints.
d. Virtualization—virtual machine (VM) separation.
e. Multitenancy policies.
5. Data quality policy:

a. a. Low-quality data may be acceptable to consumer if it provides cost saving.
6. Service compliance policy:

a. Standards adhered.
b. Certifications needed.
c. Government regulations adhered.

While we have developed ontologies for generic processes, domain-specific technical specifications will require their own ontologies. For example, for the computing service, the ontology will define the semantics of each computing term like processor speed, processor memory, number of cores, and so on. Cloud vendors may bundle their service offerings in any combination and give it brand names like "compute unit"; however, the technical specifications will specify each attribute desired and so will make it possible to query across disparate services offering similar service with different attributes bundled together. Many such ontologies exist and can be used, for example, DReggie [45]. This is part of the W3C standardized Semantic Web approach.

Fig. 2 illustrates the high-level ontology for this phase. The two main 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 functional specifications include the tasks to be automated by the service, the cloud service layer, and the service domain. The three cloud service layers that have been identified by NIST [19] are infrastructure as a service, platform as a service, and software as a service. The functional specifications also include the budgetary policies and cost (the price that the consumer is ready to pay for the service) constraints associated with the service. The technical specifications 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. The security constraints specified in the RFS include policies regarding service role/permission levels, data security policies, and cloud location/ownership policies.
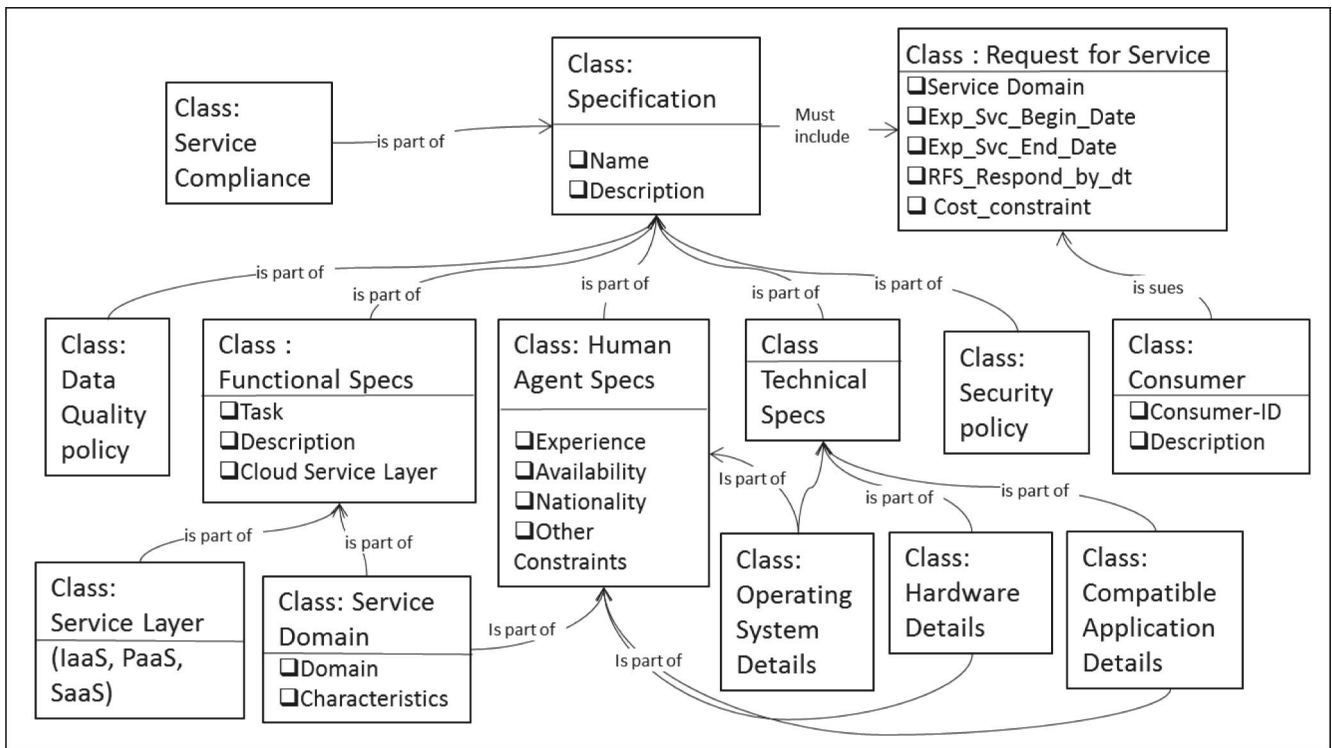
Fig. 2. The ontology of the service requirements phase contains the RFS class that includes the Specification class.

Part of our ongoing work is to use existing ontologies that have been developed for classes like standard hardware, operating systems, and computer applications. Semantic Web policy language, like AIR [13], can be used to describe service specifications and constraints in machine-processable format.

Most large organizations already have clearly defined policies for acquiring services. In addition, the policies that have to be specified in the RFS already exist as institutional or enterprise policies. These enterprise policies are centrally managed by the organization's head and may be electronically maintained across various organizational functions like Legal, Human Resources, Procurement, IT and Telecommunications, Facilities, and Security. It will be a one-time effort for the organization to consolidate these policies into a single machine readable format to create organization's service policy document or system. The service policy document can then be invoked each time a new RFS has to be issued thereby automating the RFS process. This will significantly reduce the amount of time needed for the Service Requirements phase and also significantly reduce if not completely eliminate policy oversight while acquiring a service.

3.2 Service Discovery Phase

In the service discovery phase, providers are discovered by comparing the specifications listed in the RFS with service descriptions. 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. An organization can release the RFS to a limited preapproved set of providers. Alternatively, it can search for all possible vendors on the Internet. While searching the provider, service search engines or cloud brokers can be employed. A "cloud broker" role has been identified in the

NIST reference architecture [20] which we use in our ontology. This cloud broker runs a query against the services registered with a central registry or governing body and matches the service layer, domain, data type, compliance needs, and functional and technical specifications and returns the result with the service providers matching the maximum number of requirements listed at the top. Sbodio et al. [26] and Paliwal et al. [35] have presented semantic approaches for service discovery which can be incorporated into our methodology.

One critical part of this phase is service certification, in which the consumers will contact a central registry, like UDDI [25], to get references for providers that they narrow down to. The NIST reference architecture [20] has identified a cloud auditor role that will be primarily responsible for security audit, performance audit, and privacy impact audit of the cloud. We use this role in our ontology to be the "provider certifying agent" that will be referenced in the Service Discovery Phase.

Fig. 3 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 cloud auditor validates the provider's credentials and issues a service certification if the credentials are fine. The cloud consumer's policies will determine if the cloud provider certification is essential or it can be skipped. Large organizations with stricter security policies can mandate that a provider is added to the provider's list only after the certification is received.

If the cloud 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
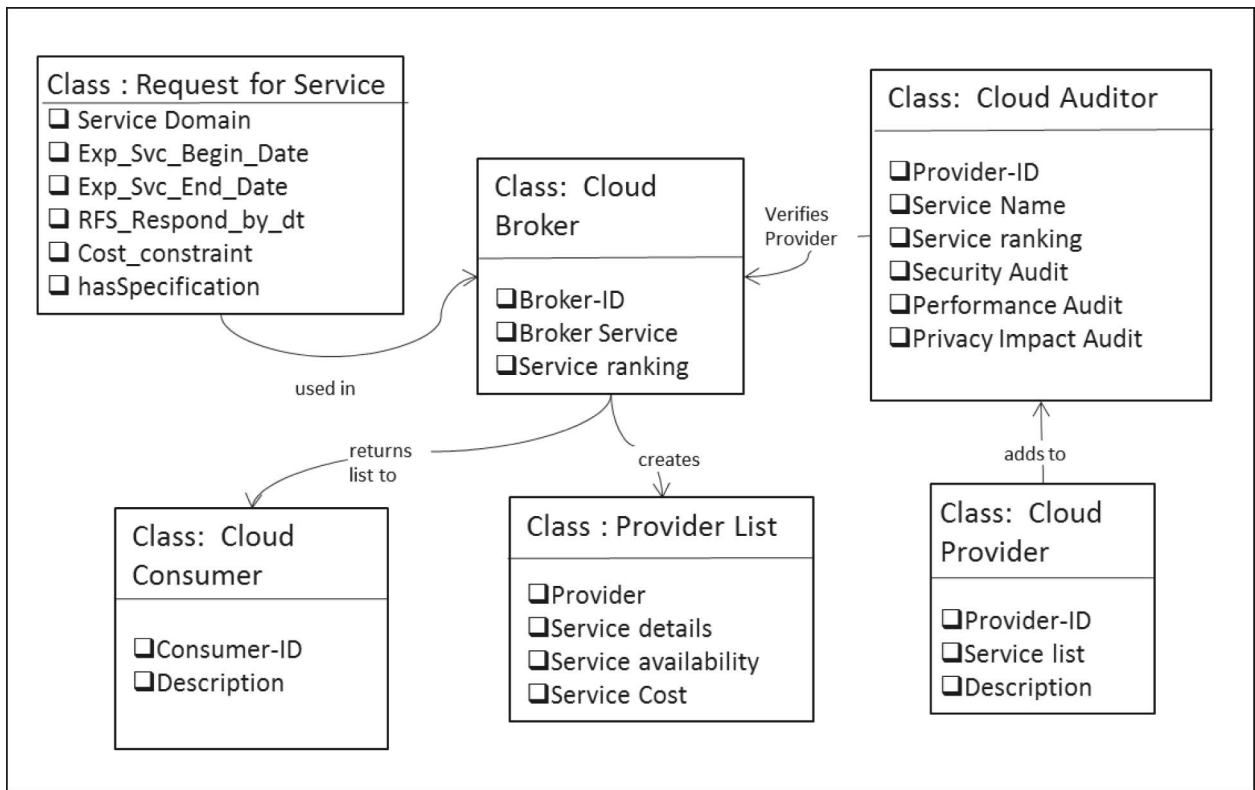
Fig. 3. The ontology for the service discovery phase uses the RFS class to search for providers and generate a provider list.

service to meet the consumer's specifications. The cloud consumer will have to begin negotiation with the service providers which is the next phase of the life cycle. Each search result will return the primary provider who will be negotiating with the consumer.

3.3 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. In our discussion with our collaborators, we found that the negotiation of SLA for the cloud services procured is the most time-consuming portion of the cloud service procurement process. Automation of this process using SPARQL queries is itself a performance improvement over the existing human-based negotiation. The service to be delivered is determined by the specifications laid down in the RFS. Service acceptance is usually guided by the service-level agreements [28] 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 Fig. 4.

Steps for Service Negotiation on the Cloud.

1. The consumer sends an RFS to the provider specifying the functional and nonfunctional requirements.
2. The provider responds to the RFS in one of three ways:

   a. Informs the consumer that it cannot provide the service, terminating negotiation.
   b. Indicates that a service matching all the requirements exists and sends the quote with SLAs.
   c. 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:

   a. 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.
   b. 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.
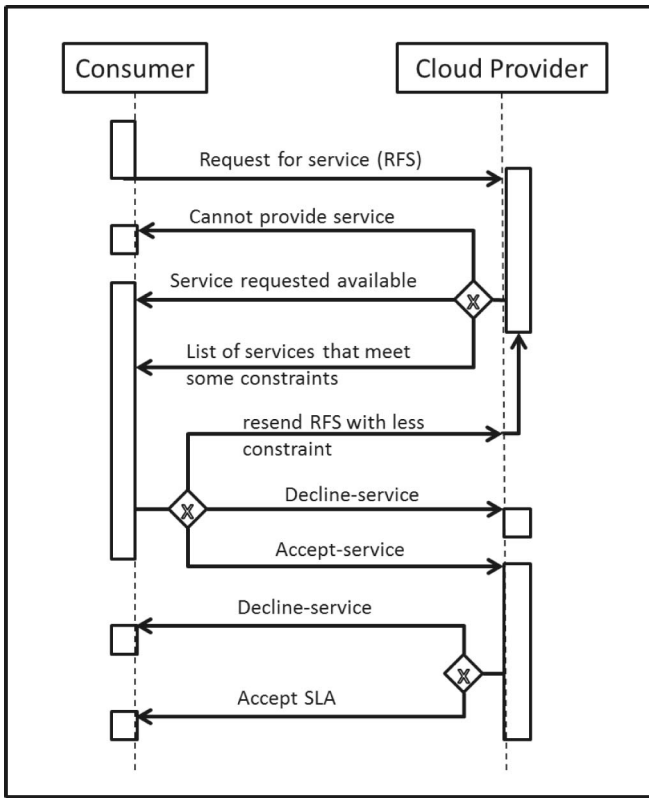   c. The consumer can decline the service, terminating the negotiation.

Fig. 4. Service negotiation sequence diagram.

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

   a. The provider can no longer provide the service, and rejects the agreement, terminating negotiation.
   b. The provider agrees with the constraints, and the same RDF file consisting of the SLA now exists with both parties.

We have constructed a high-level ontology for this phase and it is illustrated in Fig. 5. 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 subcontracts between the service provider and component (or dependent services) providers. The QoS are the essential part of the service subcontracts and are used in the consumption phase to monitor service performance.

3.4 Service Composition Phase

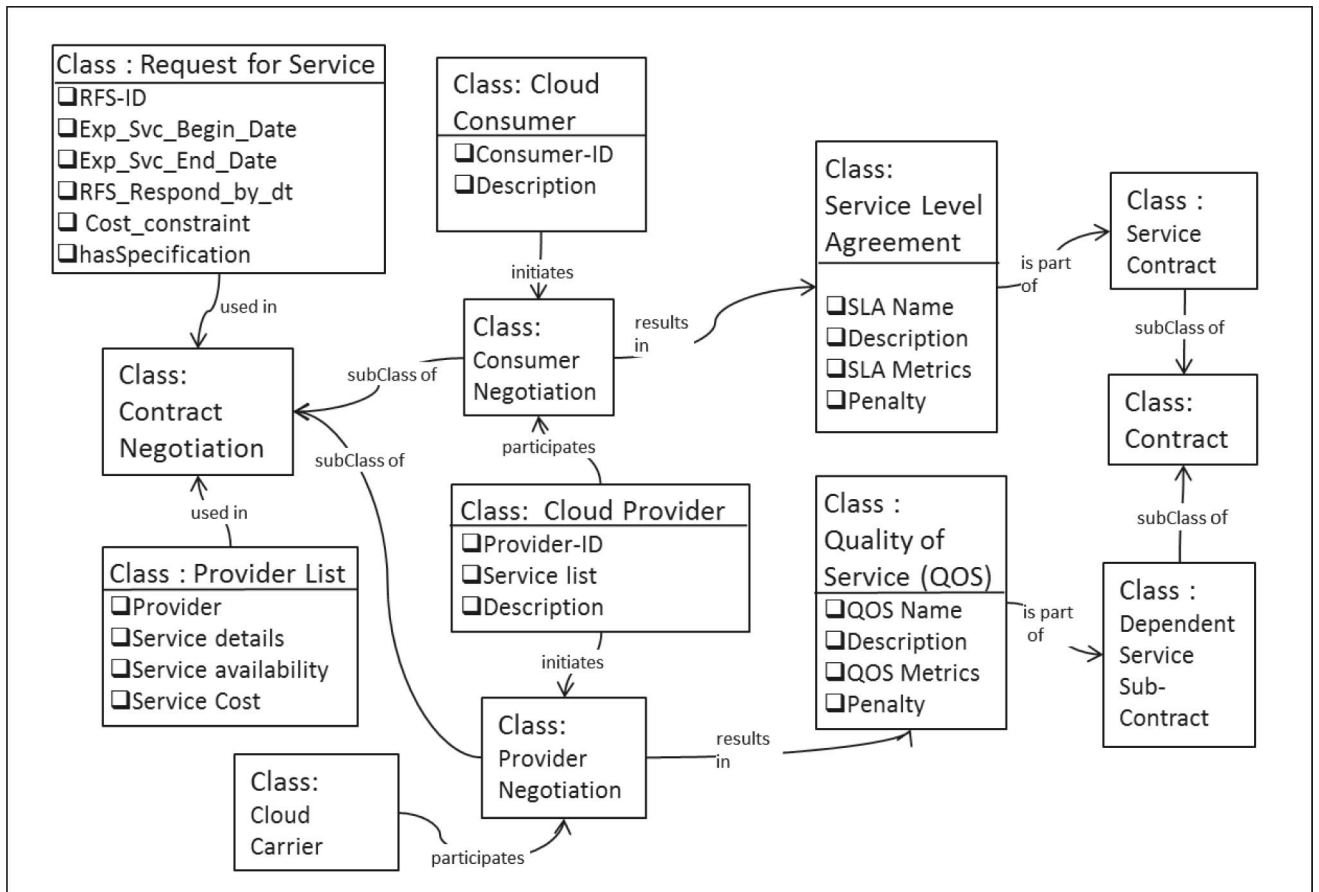In this phase, one or more components provided by one or more providers are combined and delivered as a single

Fig. 5. The ontology for service negotiation uses the RFS class for the contract negotiation and creation of the SLA and QoS.
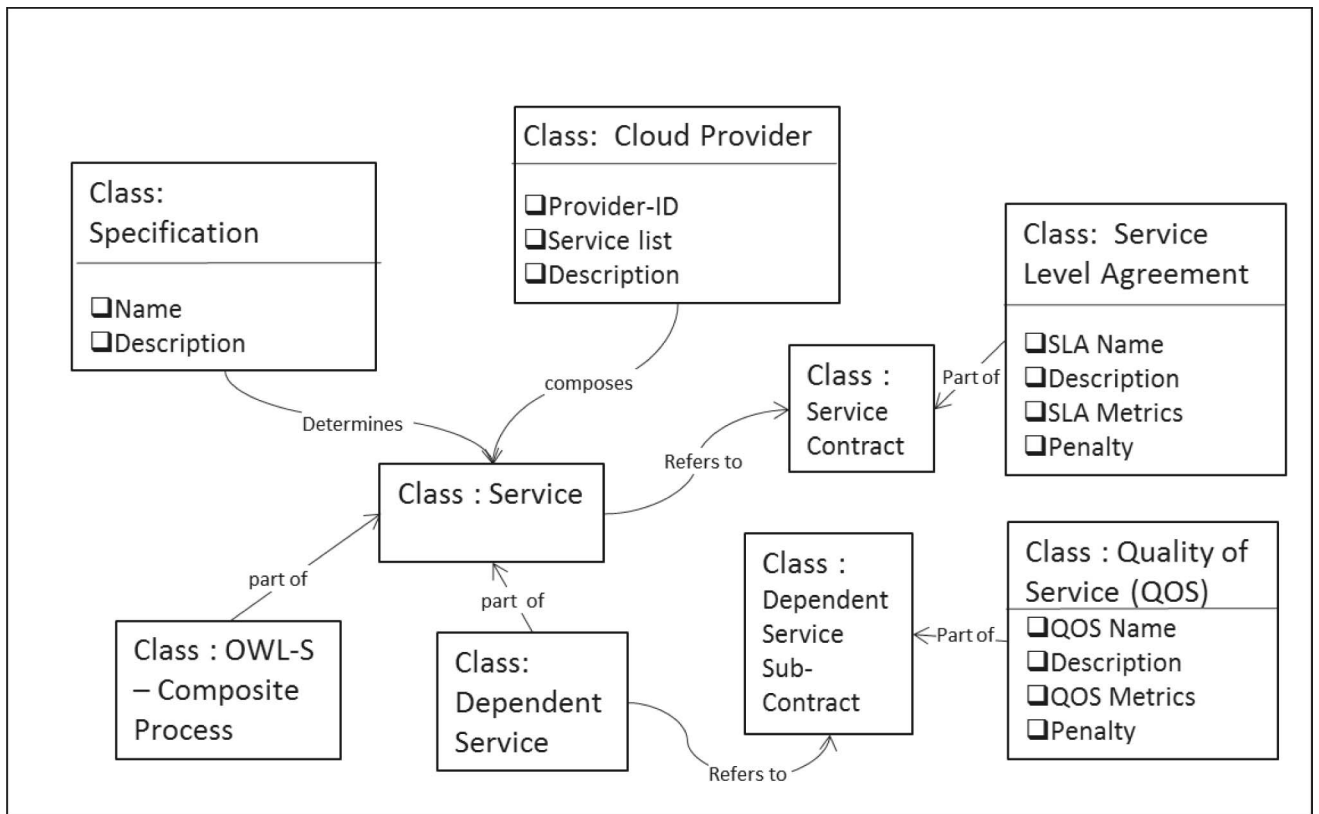
Fig. 6. The ontology for the composition phase builds on the OWL-S composite process class.

service to the service consumer. Service orchestration determines the sequence of the service components.

Fig. 6 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 determine the orchestration of the various components.

3.5 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 based on the pricing model agreed to in the SLA. 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, she/he should have the option to terminate the service and stop service payment.

Fig. 7 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. The providers have to track the service performance, reliability, assurance, and presentation as it will influence customer's satisfaction rating (CSATs). Since the dependent services/components will be at the back end 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 [12].

4 CLOUD STORAGE SERVICE PROTOTYPE

In this section, we describe the prototype that we have constructed as a proof of concept for our proposed life cycle and ontology. This prototype is based on the actual use case 3.9 [36] identified by NIST's cloud computing initiative. It demonstrates the capability that cloud users will have in the future to automatically acquire IT services from the cloud. There are many cloud providers, like Amazon or Dropbox, that provide cloud storage services. However, to show end-end operation of our system (from policy specification to service acquisition), we developed the prototype on an open source cloud platform (Eucalyptus). This let us demonstrate, for example, that the cloud provider could satisfy the user request for virtual machine separation, which was a key requirement for NIST. Our framework is fully capable of describing such constraints, which we demonstrated by using real
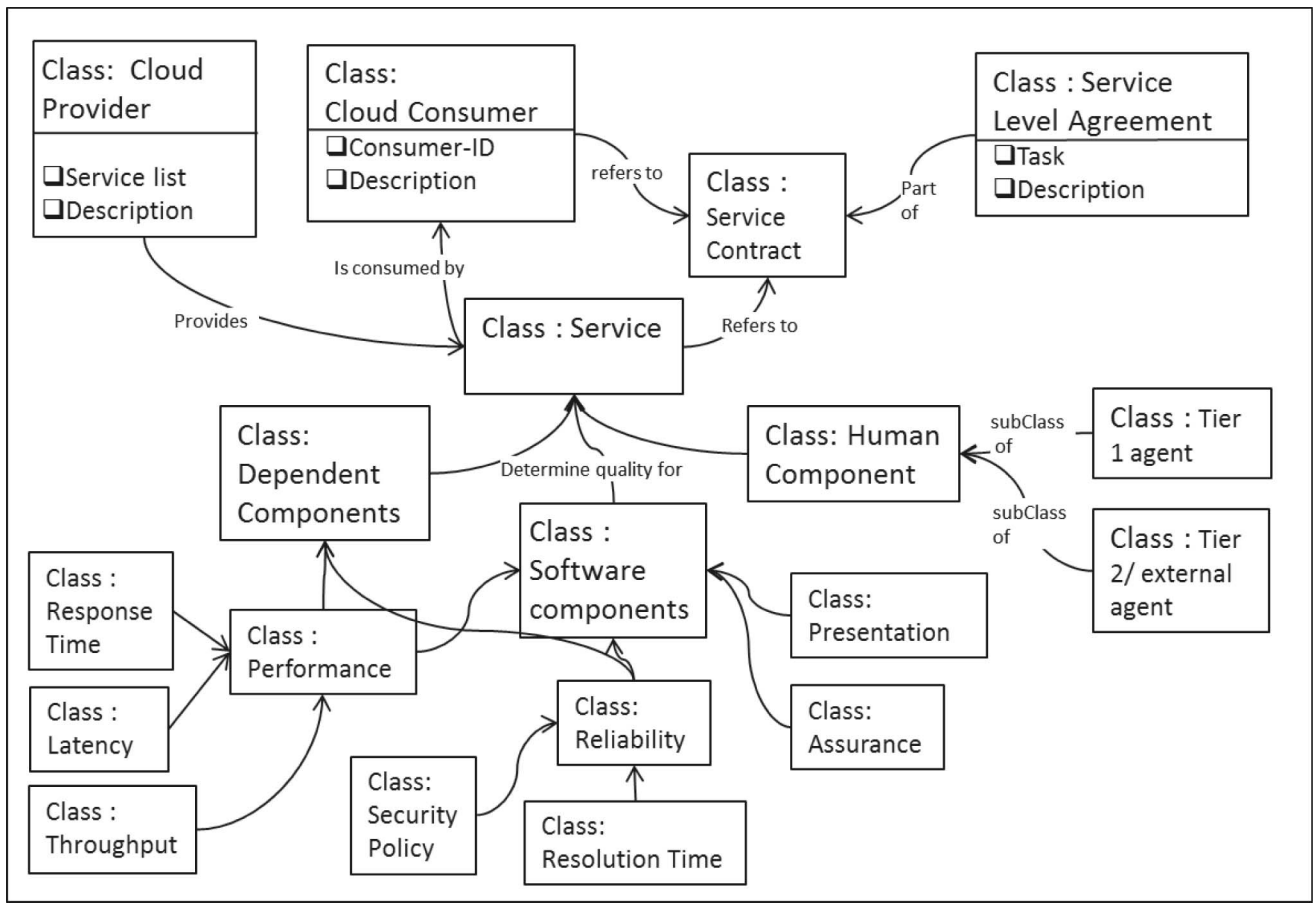
Fig. 7. The ontology for the consumption phase contains classes to monitor the quality of software, human, and dependent components of the composite process.

constraints that represent federal agency requirements that we obtained from NIST. However, there is no way for us to invoke such mechanisms on closed clouds such as Amazon or Dropbox. As such, the demonstration prototype is built on an open-source platform.

4.1 Service Description

For the prototype, we consider a simple storage service, as a representative scenario for infrastructure as a service, whereby users can store their files/data on the cloud. It consists of a web interface 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 SLAs by the process of negotiation.

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 [36] identified by the NIST cloud computing initiative. While requesting the storage service, users will specify the following service attributes depending on their storage needs:

1. Storage size needed (in GB/TB units).
2. Service cost (the price consumers are willing to pay for the service).

3. Data preservation/backup requirements (hot back-up-Yes/No; daily/weekly).
4. Service availability (e.g., 99 percent, 99.9 percent).
5. Data location (restricted to a geolocation or can be anywhere in the world).
6. Data deletion policy (data deleted or merely made inaccessible, secure wipe or not).
7. Data Encryption policy (data stored encrypted or not; encryption algorithm used, key strength).
8. Compliance policy—compliance or noncompliance for a Trusted Internet Connection (TIC) specification, CC Evaluation Assurance Level (EAL) levels.
9. User authentication mechanism (FIPS 140-2 sup-ported?).
10. Virtual machine separation (supported or not).
11. Size, speed, and number of cores for an instance specification.
12. SOAP or REST interface for a storage specification.

In addition to the NIST policies, we have also referred to service procurement policies of a large international financial organization. The main goal of this organization's service procurement policy is to acquire the "best value" service that will have an optimal combination of technical factors (like quality, functionality, service, innovation, environmental sustainability) and financial factors (like purchase price, total cost of ownership, etc.) that meet the organization's needs. To acquire the "best value" service in

Fig. 8. User interface for discovering cloud storage service by specifying constraints.

a transparent fashion, the organization's policies mandate that any purchases above US$25,000 have to be done via a competitive procurement process that considers multiple competing proposals from qualified suppliers and makes an award decision based on the merits of each proposal, relative to some predetermined criteria for best value. Exceptions are made if the service product is sold by only one vendor (sole-sourced) thereby rending the competitive bid a moot point. To continue receiving "best value" service, the service contract by policy is limited to three years and then competitively rebid at the end of the third year. Every service provider is expected to sign a service-level agreement as part of the service contract. The essential elements of the SLA include the availability time frame of service, contingency plans, time frames for notification and recovery following an unplanned service disruption or a security incident, problem resolution and escalation proce-dures, and scheduled maintenance times. We have used these elements when developing the SLAs during the negotiation phase.

4.2 Prototype Platform

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 [8], and the Joseki software [9], 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 [21] which is an open-source cloud platform that we have installed in our research lab. We are using our service life-cycle 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 another OWL ontology to describe the technical and security policies for our prototype.

4.3 Service Requirements

In the requirements phase, we identify the service layer which is IaaS for our prototype, the service domain—in this instance storage service—and the functional and technical specifications. Functional specification describe in detail what functions/tasks should the service help automate. These are mandatory attributes that the service provider must provide. For our prototype, the service attributes are the storage size, backup rules, service availability, and service costs. Specifications also list acceptable security levels, data quality, and performance levels of the service software. Service compliance details, like required certifications, standards to be adhered to, and so on, are also identified.

Our prototype has a web-based user interface, illustrated in Fig. 8, which allows consumers to generate their RFS by using drop down lists. The interface logically separates the various components of the RFS into four sections: the mandatory service attributes include constraints that have to be met, the data and security policies, compliance policies, and cloud instance. Each field has an associate "Help" description to help users determine which option to select.

```
<?xml version="1.0"?>
<rdf:RDF
xmlns="http://www.w3.org/2002/07/owl#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:itso="http://ebiquity.umbc.edu/ontologies/itso/1.0/itso.owl"
xmlns:stg="http://www.cs.umbc.edu/~kjoshi1/storage_ontology.owl"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<rdf:Description rdf:about="http://localhost/RFS">
<itso:RFS_Respond_By_Date> Sun Dec 25 14:48:27 2011
</itso:RFS_Respond_By_Date>
<itso:Expected_Begin_Date_of_Service> 1-1-2012
</itso:Expected_Begin_Date_of_Service>
<itso:Service_Cost_Constraint> 0 </itso:Service_Cost_Constraint>
<itso:Service_Location_constraint> global
</itso:Service_Location_constraint>
<stg:storage> 2GB </stg:storage>
<stg:backup> Weekly </stg:backup>
<stg:availability> 95 </stg:availability>
<stg:datadeletion> data archived </stg:datadeletion>
<stg:Encryption> No Encryption </stg:Encryption>
<stg:authentication> FIPS 140 2 supported </stg:authentication>
<stg:VMseparation> Not needed </stg:VMseparation>
<stg:storage_interface> SOAP WSDL </stg:storage_interface>
<stg:TIC_connection> TIC Compliant </stg:TIC_connection>
<stg:CC_EAL> 3 </stg:CC_EAL>
<stg:cloud_instance_size> 1GB </stg:cloud_instance_size>
<stg:cloud_instance_speed> 1GHz </stg:cloud_instance_speed>
<stg:cloud_instance_cores> 6 </stg:cloud_instance_cores>
</rdf:Description>
</rdf:RDF>
```

Fig. 9. RFS generated as an RDF/XML file.

After selecting the values of their service attributes, security policies, and compliance policies, the consumers can press the "Request for Service" button to generate an RDF document that contains the RFS. Fig. 9 illustrates the RDF/XML document generated for the attributes selected in Fig. 8.

4.4 Service Discovery

The users can press the "Discover Services" button to search for services that match the RFS issued. The tool generates federated SPARQL queries, like the one illustrated in Fig. 10, based on the selections on the screen. This query runs across multiple SPARQL endpoints to retrieve a list of matching services residing on that endpoint. Researchers like Sbodio et al. [26] have also proposed algorithms for service discovery using SPARQL language.

If a query matching all the constraints is found, it is displayed on the screen. Else the user is advised to begin service negotiation by selecting the Negotiation button.

4.5 Service Negotiation

The users can press the "Negotiate and Finalize SLA" button to begin the service negotiation. The tool automatically begins relaxing RFS constraints one by one by removing the constraints from the SPARQL query and generating a new SPARQL query to search the endpoints.

```
PREFIX stg:    http://www.cs.umbc.edu/~kjoshi1/storage_ontology.owl
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>
SELECT ?serviceName ?textDescription ?Cost ?creator ?Backup ?Availability
?Storage_size ?datadeletion ?Encryption ?authentication ?VMseparation ?stor-
age_interface ?TIC_connection ?CC_EAL ?cloud_instance_size
?cloud_instance_speed ?cloud_instance_cores
  { SERVICE <http://eb4.cs.umbc.edu:2020/Storage>
    { SELECT  ?serviceName ?textDescription ?creator ?Cost ?Location ?Backup
?Availability ?Storage_size ?datadeletion ?Encryption ?authentication ?VMsepa-
ration ?storage_interface ?TIC_connection ?CC_EAL ?cloud_instance_size
?cloud_instance_speed ?cloud_instance_cores
      WHERE
      {?serviceName owls:textDescription ?textDescription .
       ?serviceName sv:creator ?creator . ?serviceName stg:Storage_size ?Stor-
age_size FILTER regex(?Storage_size, "2GB" ,"i").
?serviceName stg:Cost ?Cost FILTER regex(?Cost, "0" ,"i").
?serviceName stg:Availability ?Availability FILTER regex(?Availability, "95"
,"i").
?serviceName stg:Backup ?Backup FILTER regex(?Backup, "Weekly","i").
?serviceName stg:authentication ?authentication FILTER regex(?authentication,
"FIPS 140 2 supported" ,"i").
?serviceName stg:Encryption ?Encryption FILTER regex(?Encryption, "No
Encryption" ,"i").
?serviceName stg:Location ?Location FILTER regex(?Location, "global" ,"i").
?serviceName stg:datadeletion ?datadeletion FILTER regex(?datadeletion, "data
archived" ,"i").
?serviceName stg:VMseparation ?VMseparation FILTER regex(?VMseparation,
"Not needed" ,"i").
?serviceName stg:storage_interface ?storage_interface FILTER regex (?stor-
age_interface, "SOAP WSDL" ,"i").
?serviceName stg:TIC_connection ?TIC_connection FILTER regex
(?TIC_connection, "TIC Compliant" ,"i").
?serviceName stg:CC_EAL ?CC_EAL FILTER regex(?CC_EAL, "3" ,"i").
?serviceName stg:cloud_instance_size ?cloud_instance_size FILTER re-
gex(?cloud_instance_size, "1GB" ,"i").
?serviceName stg:cloud_instance_speed ?cloud_instance_speed FILTER re-
gex(?cloud_instance_speed, "1GHz" ,"i").
?serviceName stg:cloud_instance_cores ?cloud_instance_cores FILTER re-
gex(?cloud_instance_cores, "10" ,"i").
  } } }
```

Fig. 10. Service discovery by using the SPARQL query to get a service description.

The order of constraints 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 a 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. The consumer can finalize the SLA by accepting the service that best matched the constraints. The final SLA is saved as an RDF file and is in machine readable format.

4.6 Service Composition and Consumption

The user tool is interfaced to the Eucalyptus [21] cloud, which is an infrastructure as a service cloud solution. The tool and the Eucalyptus cloud were installed on separate machines. Due to security reasons, the Eucalyptus

installation had no direct internet access and no direct access to the tool. The two systems communicated through an intermediate node called Bluegrit which is a 116 core PowerPC cluster managed at UMBC.

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. The 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.

5 CONCLUSION AND ONGOING WORK

In this paper, we have defined an integrated ontology for processes needed to automate IT services life cycle 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.

The tool that we built successfully demonstrated how our methodology can be used to significantly 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, we are working on automating complex service negotiation process where the negotiation is on a range of values for a constraint. We are also updating and refining the ontology to capture these complex negotiation protocols that we are designing. 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) [37] platform provided by IBM. We also plan on using enterprise policies from various organizations to demonstrate the validity of this framework.

ACKNOWLEDGMENTS

The authors wish to thank Ms. Dawn Leaf, Program Director of the Cloud Computing Initiative at NIST, for her support for this work.

REFERENCES

[1] F. Baader, D. Calvanese, D. McGuinness, P. Patel-Schneider, and D. Nardi, *The Description Logic Handbook: Theory, Implementation, and Applications.* Cambridge Univ. Press, 2003.
[2] 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,* vol. 8, no. 3, pp. 249-269, 2008.
[3] D. Bianchini, V. De Antonellis, B. Pernici, and P. Plebani, "Ontology-Based Methodology for E-Service Discovery," *Int'l J. Information Systems,* Special Issue: The Semantic Web and Web Services, vol. 31, nos. 4/5, pp. 361-380, June/July 2006.
[4] J. Black et al., "An Integration Model for Organizing IT Service Management," *IBM Systems J.,* vol. 46, no. 3, pp. 405-422, 2007.
[5] B. Boehm, "A Spiral Model of Software Development and Enhancement," *ACM SIGSOFT Software Eng. Notes,* vol. 11, no. 4, pp. 14-24, Aug. 1986.
[6] M. Hepp, "GoodRelations: An Ontology for Describing Products and Services Offers on the Web," *Proc. 16th Int'l Conf. Knowledge Eng. and Knowledge Management (EKAW '08),* pp. 332-347, 2008.
[7] 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, 2004.
[8] Apache Software Foundation, "Jena - A Semantic Web Framework for Java," http://incubator.apache.org/jena/, Mar. 2012.
[9] "Joseki - A SPARQL Server for Jena," http://www.joseki.org/, Mar. 2012.
[10] K. Joshi, T. Finin, and Y. Yesha, "Integrated Lifecycle of IT Services in a Cloud Environment," *Proc. Third Int'l Conf. Virtual Computing Initiative (ICVCI '09),* Oct. 2009.
[11] K. Joshi, "OWL Ontology for Lifecycle of IT Services on the Cloud," http://ebiquity.umbc.edu/ontologies/itso/1.0/itso.owl, 2010.
[12] K. Joshi, A. Joshi, and Y. Yesha, "Managing the Quality of Virtualized Services," *Proc. SRII Global Conf.,* Mar. 2011.
[13] L. Kagal, C. Hanson, and D. Weitzner, "Using Dependency Tracking to Provide Explanations for Policy Management," *Proc. IEEE Int'l Workshop Policies for Distributed Systems and Networks,* 2008.
[14] J. Kopecky, T. Vitvar, C. Bournez, and J. Farrell, "SAWSDL: Semantic Annotations for WSDL and XML Schema," *IEEE Internet Computing,* vol. 11, no. 6, pp. 60-67, Nov./Dec. 2007.
[15] O. Lassila et al., "Resource Description Framework (RDF) Model and Syntax Specification," W3C recommendation, 1999.
[16] D. Martin et al., "Bringing Semantics to Web Services: The OWL-S Approach," *Lecture Notes in Computer Science,* vol. 3387, pp. 26-42, Springer, 2005.
[17] E.M. Maximilien and M. Singh, "A Framework and Ontology for Dynamic Web Services Selection," *IEEE Internet Computing,* vol. 8, no. 5, pp. 84-93, Sept./Oct. 2004.
[18] D. McGuinness et al., "OWL Web Ontology Language Overview," W3C recommendation, 2004.
[19] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," National Institute of Standards and Technology Special Publication 800-145, Sept. 2011.
[20] F. Liu et al., "NIST Cloud Computing Reference Architecture," National Institute of Standards and Technology Special Publication 500-292, Nov. 2011.
[21] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The Eucalyptus Open-Source Cloud-Computing System," *Proc. Ninth IEEE/ACM Int'l Symp. Cluster Computing and the Grid,* pp. 124-131, 2009.
[22] M. Papazoglou and W. Van Den Heuvel, "Service-Oriented Design and Development Methodology," *Int'l J. Web Eng. and Technology,* vol. 2, no. 4, pp. 412-442, 2006.
[23] S. Paurobally, V. Tamma, and M. Wooldrdige, "A Framework for Web Service Negotiation," *ACM Trans. Autonomous and Adaptive Systems,* vol. 2, no. 4, article 14, Nov. 2007.
[24] E. Prud'hommeaux and A. Seaborne, "SPARQL Query Language for RDF," W3C recommendation, http://www.w3.org/TR/rdf-sparql-query/, Jan. 2008.
[25] S. Ran, "A Model for Web Services Discovery with QoS," *ACM SIGecom Exchanges,* vol. 4, no. 1, pp. 1-10, 2003.
[26] M.L. Sbodio, D. Martin, and C. Moulin, "Discovering Semantic Web Services Using SPARQL and Intelligent Agents," *J. Web Semantics,* vol. 8, no. 4, pp. 310-328, Nov. 2010.
[27] A. Sheth, K. Gomadam, and A. Ranabahu, "Semantics Enhanced Services: METEOR-S, SAWSDL and SA-REST," *IEEE Data Eng. Bull.,* vol. 31, no. 3, pp. 8-12, 2008.
[28] SLA@SOI, "What's in a Service Level Agreement?" http://sla-at-oi.eu/2009/03/what's-in-a-service-level-agreement/, Mar. 2012.
[29] R. Smith, "The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver," *IEEE Trans. Computers,* vol. 29, no. 12, pp. 1104-1113, Dec. 1980.
[30] Semantic Web, "SPARQL Endpoint," http://semanticweb.org/wiki/SPARQL_endpoint, Mar. 2012.
[31] J.V. Bon et al., *Foundations of IT Service Management Based on ITIL V3.* Van Haten, 2008.

[32] G. Williams, "SPARQL 1.1 Service Description," http://www.w3.org/TR/2009/WD-sparql11-service-description-20091022/, 2013.

[33] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q. Sheng, "Quality Driven Web Services Composition," *Proc. 12th Int'l Conf. World Wide Web,* pp. 411-421, 2003.

[34] K. Ren, N. Xiao, and J. Chen, "Building Quick Service Query List Using WordNet and Multiple Heterogeneous Ontologies toward More Realistic Service Composition," *IEEE Trans. Services Computing,* vol. 4, no. 3, pp. 216-229, July/Sept. 2011.

[35] A. Paliwal, B. Shafiq, J. Vaidya, H. Xiong, and N. Adam, "Semantics Based Automated Service Discovery," *IEEE Trans. Services Computing,* vol. 5, no. 2, pp. 260-275, Apr.-June 2012.

[36] "Cloud Computing Use Case 3.9: Query Cloud-Provider Capabilities and Capacities," National Institute of Standards and Technology working document, http://www.nist.gov/itl/cloud/3_9.cfm, Nov. 2010.

[37] J. Moothoor and V.A. Bhatt, "A Cloud Computing Solution in Universities," IBM developerWorks technical document, http://www.ibm.com/developerworks/webservices/library/ws-vcl/, Jan. 2012.

[38] M. Xu, Z. Hu, W. Long, and W. Liu, "Service Virtualization: Infrastructure and Applications," *The Grid: Blueprint for a New Computing Infrastructure,* I. Foster and C. Kesselman, eds., Morgan Kaufmann, 2004.

[39] D. De Roure et al., "The Semantic Grid: Past, Present, and Future," *Proc. IEEE,* vol. 93, no. 3, pp. 669-681, Mar. 2005.

[40] T. Dörnemann, E. Juhnke, and B. Freisleben, "On-Demand Resource Provisioning for BPEL Workflows Using Amazon's Elastic Compute Cloud," *Proc. Ninth IEEE/ACM Int'l Symp. Cluster Computing and the Grid (CCGRID '09),* 2009.

[41] C. Crawford, G. Bate, L. Cherbakov, K. Holley, and C. Tsocanos, "Toward an on Demand Service-Oriented Architecture," *IBM Systems J.,* vol. 44, no. 1, pp. 81-107, 2005.

[42] Q.Z. Sheng et al., "Configurable Composition and Adaptive Provisioning of Web Services," *IEEE Trans. Services Computing,* vol. 2, no. 1, pp. 34-49, Jan.-Mar. 2009.

[43] M. Boniface et al., "Dynamic Service Provisioning Using GRIA SLAs," *Proc. Int'l Conf. Service-Oriented Computing (ICSOC '07),* 2007.

[44] A.J. Ferrer et al., "OPTIMIS: A Holistic Approach to Cloud Service Provisioning," *Future Generation Computer Systems,* vol. 28, pp. 66-77, 2011.

[45] D. Chakraborty, F. Perich, S. Avancha, and A. Joshi, "DReggie: Semantic Service Discovery for M-Commerce Applications," *Proc. Workshop Reliable and Secure Applications in Mobile Environment in Conjunction with 20th Symp. Reliable Distributed Systems,* 2001.

[46] L. Badger et al., "US Government Cloud Computing, Technology Roadmap, Volume I, Release 1.0 (Draft)," National Institute of Standards and Technology Special Publication 500-293, http://www.nist.gov/itl/cloud/upload/SP_500_293_volumeI-2.pdf, Nov. 2011.

Karuna P. Joshi received the bachelors degree in computer engineering from the University of Mumbai, India, and the MS and PhD degrees in computer science from the University of Maryland Baltimore County (UMBC), where she was twice awarded the IBM PhD Fellowship. She is now a research assistant professor at UMBC whose primary research area is cloud computing and healthcare IT. She has developed a framework to automate the acquisition and consumption of cloud-based services. She is currently researching cloud security and privacy issues, especially related to healthcare services. She is also working in collaboration with NIST to develop standards for cloud usability. She has more than 20 years of experience, primarily as a cloud researcher and an IT manager. She worked at the International Monetary Fund for nearly a decade. Her managerial experience includes portfolio management, program/project management, and change management.

Yelena Yesha received the BSc degree in computer science from York University, Toronto, Canada, and the MSc and PhD degrees in computer and information science from the Ohio State University. She is currently a professor of computer science and electrical engineering at the University of Maryland, Baltimore County. She is also the associate director for the Multicore Computational Center. In addition, she served as the director of the Center of Excellence in Space Data and Information Sciences at NASA. Her research interests are in the areas of distributed databases, distributed systems, digital libraries, electronic commerce, and trusted information systems. She has coauthored 14 books and authored more than 180 refereed articles in these areas.
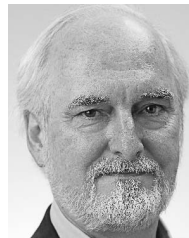
Tim Finin holds degrees from MIT and the University of Illinois and has also held positions at Unisys, the University of Pennsylvania, and the MIT AI Laboratory. He is a professor of computer science and electrical engineering at University of Maryland, Baltimore County. He has more than 30 years of experience in applications of artificial intelligence to problems in information systems and language understanding. His current research focuses on the Semantic Web, mobile computing, analyzing and extracting information from text and online social media, and enhancing security and privacy in information systems.