Asian Journal of Information Technology 15 (23): 4900-4915, 2016
ISSN: 1682-3915
© Medwell Journals, 2016

Ontology-Based Saas Catalogue for Cloud Services Publication and Discovery

Yasmine M. Afify, Ibrahim F. Moawad, Nagwa L. Badr and M. F. Tolba
Faculty of Computer and Information Sciences, Ain Shams University, 11566 Cairo, Egypt

Abstract: The number of software providers offering their applications as a Software-as-a-Service (SaaS) to exploit the benefits of cloud computing is increasing. New challenges to the cloud services discovery are imposed due to the SaaS services unique characteristics such as various and dynamic service offerings and the lack of standard description language. In this study, we propose OntSaaS, an ontology-based system for SaaS publication and discovery. Ont SaaS standardizes the advertisement process, serves as a semantic-based catalogue for the service offerings and provides competent search capabilities for the user. The main building blocks of the proposed system are the unified SaaS ontology and a semantic business-oriented matchmaking technique that maps requests and offers of cloud SaaS services. The proposed request-service matchmaking algorithm merges semantic-based services metadata with ontology-based hierarchical matching. Prototypical implementation and evaluation of the system proved its performance enhancement in respect of the service utility and success rate. Results showed that the concept recommendation approach employed decreased the service registration process time. Moreover, the proposed matchmaking algorithm similarity results revealed the actual relevance of the offered services to the user requests. Finally, the proposed ontology-based expansion approach for the user request improved the user opportunity to find appropriate services to his requirements in case of discovery partial match.

Key words: Cloud SaaS, service ontology, service discovery, concept recommendation, semantic annotation, matchmaking algorithm

INTRODUCTION

The SaaS model has gained widespread adoption in recent years. Therefore, deployment of substantial cloud services is greatly expected (Limam and Boutaba, 2010; Yu, 2015). Consequently it is important to assist users to find their desired service (Kang and Sim, 2016). SaaS Service discovery is the process of searching for services, with functional and non-functional characteristics that satisfy user requirements. At present, there is no standard protocol or search mechanism for discovering cloud services.

To discover cloud services, users have to perform the search based on their own knowledge and key words. However, manual search using traditional search engines is a tedious and time consuming process that hinders efficient use of cloud services (Nabeeh *et al*., 2015). Moreover, cloud providers usually publish their cloud services on their portals using different formats using their own terms, in unstructured plain text. As a result, sometimes it is difficult for the user to discover the desired service information (Noor *et al*., 2013; Reshmy and Srivatsa, 2005) and to decide which service can fulfill his requirements (Garg *et al*., 2013; Chen *et al*., 2011; Zhao *et al*., 2012; Tserpes *et al*., 2012). Additionally, service plans and offers may be updated any time without any prior notifications which requires continuous follow-up from the user side.

As a result, some specific online directories for cloud services search emerged such as: Cloudbook Cloudbook. http://www.cloudbook.net/directories/ product-services /cloud-computing-directory),Cloud Taxonomy (Open Crowd Project. http://cloudtaxonomy. opencrowd. com/taxonomy/) and CloudSurfing CloudSurfing. http:// www. cloudsurfing.com/). However, these directories suffer from the limitation of search capabilities. For efficient services discovery on the cloud it is necessary to provide sufficient and clear information on service features and characteristics in addition to quality information in a standardized form. A cloud service catalogue with a common data model is vital for the evolution of an open cloud marketplace. Hence, several service registries have emerged to link providers and consumers. However, SaaS service offerings have different characteristics than services they were designed to handle. Moreover, they suffer from limited search capabilities.

Corresponding Author: Yasmine M. Afify, Faculty of Computer and Information Sciences, Ain Shams University, 11566 Cairo, Egypt

To supplement the existing efforts, this research exploits the business perspective of the SaaS services in order to eliminate the lack of standardization problem which is considered the most significant challenge facing cloud services discovery. Besides it matches specific service functionalities required by the user to features supported by published SaaS services.

In the research (Afify *et al*., 2013, 2014a, b), we presented a semantic based system for SaaS services publication, discovery and selection. We introduced a unified SaaS ontology used for storage and retrieval of real SaaS offers. Based on this ontology, we introduced a hybrid matchmaking algorithm for SaaS services. In this study, we propose radical changes to the discovery process. Specifically, there are three contributions in this study. Firstly it introduces a user-controlled business-oriented services discovery approach in order to increase the efficiency of the discovery process. Secondly it proposes an ontology-based expansion approach in case of discovery partial matching. Thirdly it presents a hybrid algorithm for matchmaking the user request to service advertisements.

Literature review

Cloud service publication: Universal Description, Discovery and Integration (UDDI) is a registry-based approach for services publishing and querying based on Web Service Description Language (WSDL) documents. UDDI has the following limitations services are organized with respect to pre-defined categories not with respect to what they actually provide its syntactic service discovery capability is rather limited and the lack of support for non-functional properties.

An ontology-based catalog which describes the cloud computing resources offered by heterogeneous cloud providers was proposed in (Bernstein and Vij, 2010). However it only focuses on infrastructure resource capabilities and features such as CPU, storage and compliance.

Unified business service and cloud ontology with service querying were proposed in (Tahamtan *et al*., 2012) which helps users to find cloud services according to functional and non-functional requirements. Limitations of this research are the exact matching between user query and Business Functions (BF) and the query representation language which greatly limits its use to experienced users only.

A semantic registry of cloud services was proposed by Mindruta and Fortis (2013). The researchers focus was ontological support related to the semantic discovery of cloud services and their associated artifacts. Their focus was on IaaS and PaaS service models. An extensible Everything-as-a-Service (XaaS) registration entry was proposed by Spillner and Schill. They proposed an extensible description language for services, a registration model, a system for registration and subsequent service discovery operations. However, no details were given on the request-service matchmaking algorithm.

Other service registries were introduced like: Membrane SOA registry, service-finder and depot. However, they are dedicated to the services described using WSDL files. Examples of proprietary registries are: IBM WebSphere service registry (IBM WebSphere Service Registry. http://www-01.ibm.com/software/integration/wsrr/.) and Oracle service registry (Oracle Service Registry.http://www.oracle. com/ technetwork/ middleware/registry/overview/index.html). In conclusion, the following set of limitations can be highlighted:

- There is no domain specific standard for describing SaaS services
- Existing works mostly deal with resource and technical issues in cloud domain
- Business aspect of cloud services has not been appropriately utilized in existing

By considering these limitations, the proposed system is considered vital. For the cloud consumer it offers him detailed information about the functionalities supported by SaaS services from different providers. For the cloud provider it increases his service reachability.

Cloud service discovery: System architecture for automated SaaS services discovery and selection was presented by Sukkar. The system recommends service options to users based on functional and non-functional properties of the SaaS services. However, the service characteristics were not considered in the recommendation process.

Dastjerdi *et al*. (2010) presented a flexible approach for ontology-based discovery of cloud virtual units to provide QoS aware deployment of appliances on cloud service providers. However, the architecture applies on Infrastructure-as-a-Service (IaaS) services only.

A clustering and recommendation methods for Semantic Web Services (SWS) in evolution were presented by Lei *et al*. (2014). Researchers used clustering to group semantic services according to topic, functionality and description. Moreover, they presented a recommendation method for composite services that utilizes matrix decomposition. However it is specific to SWS, thus special characteristics of cloud services were not considered in the recommendation process. Some agent-based discovery systems were proposed.

Ontology-based agent generation framework for information retrieval on cloud environment was presented by Chang *et al.* (2011). It assists users to automatically generate mobile agents for discovering services. On the other hand, the researchers of Sim (2012) proposed a cloud-focused semantic search engine for cloud service discovery, called cloudle. Cloud ontology is consulted to verify similarities between service specifications and user requirements. However, the services business perspective has not been considered in the reasoning.

Chen proposed adding semantics to service description for improved cloud service discovery. WSDL constructs are mapped to domain ontology concepts and stored in the UDDI to be used for querying. However, their approach is specific to WSDL-described services which is not the case for most of the SaaS services. Moreover it is limited to exact matching of query-service ontology concepts.

A framework for a semantic service search engine that retrieves services based on domain-specific Quality of Service (QoS) criteria in the digital ecosystem environment was presented by Dong *et al.* (2011a, b). The ultimate goal of this search engine is to allow service users to retrieve and evaluate services published by the service providers. The researchers also presented a framework for a service concept recommendation system in Dong *et al.* (2011c), in which they used concept recommendation to correctly represent service requests in a semantic service matchmaker as well as a semantic similarity model for service ontology environment. The authors extended their research in Dong *et al.* (2013) where they presented a systematic framework for online service advertising information search. However it is based on service advertising information only, without taking into account its feature details.

Noor *et al.* (2013) developed a cloud services crawler engine. The collected cloud services can be continuously updated for effective cloud services discovery. However, no details were given on the discovery or matchmaking mechanism.

Lin *et al.* (2013) investigated a QoS-aware service discovery method over a service-registering enabled P2P network. A peer node registers its information to its neighbors, then at service discovery phase, the QoS-aware service discovery is supported in a probabilistic flooding way according to the network traffic. However it is limited to discovery of web service resources. A semantically-enhanced platform that assists in the process of discovering the cloud services that best match user needs was proposed in Garcia *et al.* (2014). However, the semantic-based matching process of user query relied on the service descriptions only without taking into account any information about their functionalities.

Modica and Tomarchio (2015) a semantic discovery framework was presented that assists providers and consumers of cloud services to operate in a cloud market. A semantic model is proposed that addresses the business aspects of the supply-demand matchmaking. Although many cloud service features have been considered, they were not related to service offer functionalities. To summarize, a set of limitations can be highlighted as follows:

- Most of existing systems neglect the special feature requirements for cloud users
- Existing systems do not help users refine their queries if there is no feasible result

To overcome these limitations, the proposed system provides semantic-based business-oriented discovery of SaaS services which relies heavily on matchmaking services functional metadata. Moreover, it suggests ontology-based expansion alternatives to the user when his query cannot be accurately matched to SaaS advertisements.

METERIALS AND METHODS

The proposed system architecture: OntSaaS is a Software-as-a-Service (SaaS) service publication and discovery system that employs semantic approaches to guarantee uniform representation for services and to assist the user in finding required service. As presented in Fig. 1, OntSaaS consists of four subsystems in addition to the unified SaaS ontology and the word net (Miller, 1995).

The service publication subsystem is responsible for accepting new service registrations from the cloud providers. During the registration process, the system recommends concept BFs to the cloud provider to describe the service functionalities. These concepts are retrieved from the SaaS services domain ontology. Some ontology concepts are included as metadata in the new service publication to be used later in the service discovery process. Comprehensive details on the registration template and methodology in our research (Afify *et al.*, 2014b). The service discovery subsystem is responsible for searching for services that match the user request. In order to solve the vagueness and heterogeneity problems, the concept recommendation approach (Dong *et al.*, 2011b, c; Dong *et al.*, 2013) is employed in order to find the best service that fulfills his requirements. In particular the proposed discovery

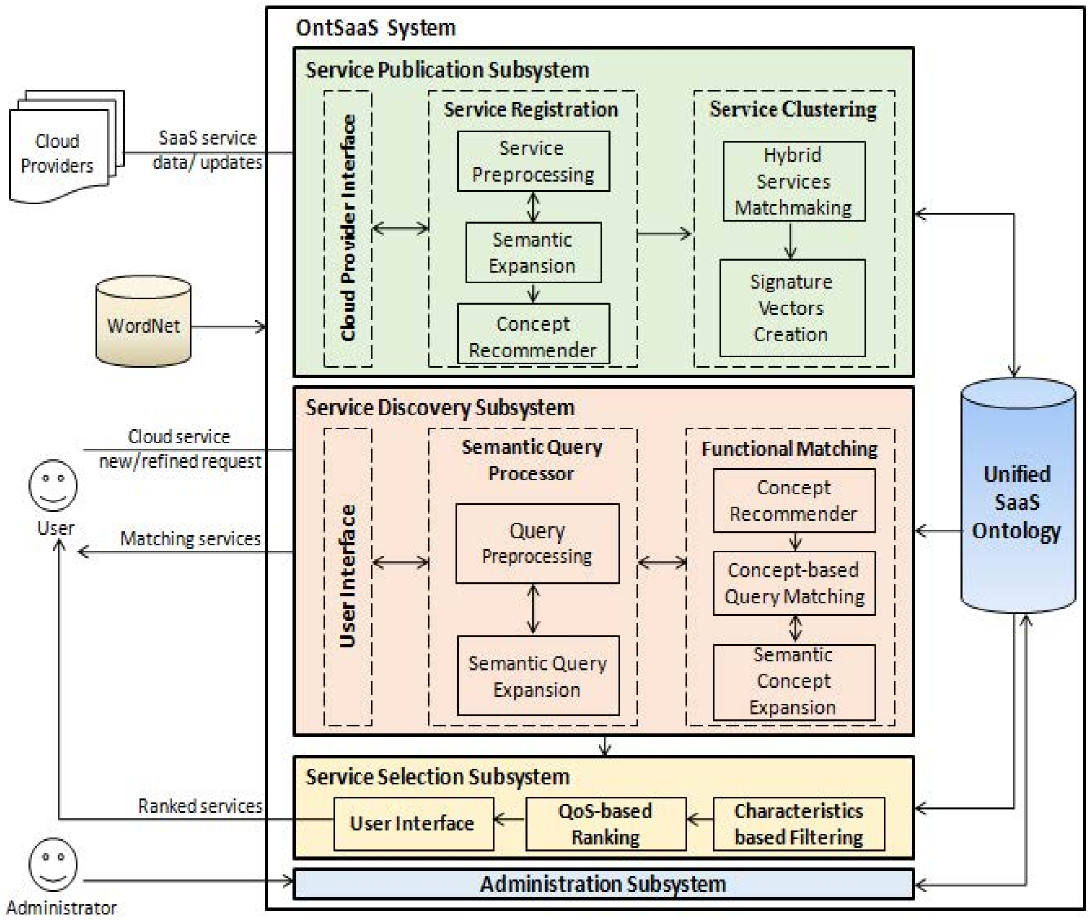


Fig. 1: OntSaaS system architecture

system utilizes some Information Retrieval (IR) methods to improve its efficiency such as the query expansion and the relevance feedback methods (Salton and McGill, 1986).

The service selection subsystem is responsible for non-functional filtering and ranking of the discovered services based on their characteristics and QoS values. More details of the selection workflow can be found in the research (Afify *et al*., 2014a). The administration subsystem is responsible for managing the whole system workflow. For example it manages the SaaS service template, enriches the services domain ontology, sets the similarity weights, etc.

The SaaS ontology is one of the core building blocks of OntSaaS. It integrates knowledge about SaaS services domain, characteristics and QoS metrics in addition to real offers. It represents a structured schema for storing both functional service capabilities and non-functional service quality guarantees. Taking into consideration the shortcomings of developing a new ontology from scratch, we have carefully analyzed the existing ontologies (Born *et al*., 2008; Youseff *et al*., 2008; Fortis *et al*., 2012; Joshi *et al*., 2014; Moscato *et al*., 2011; Hofer and Karagiannis, 2011; Hepp, 2006). It is worth to note that most of the existing ontologies serve as taxonomies and focus on technical and infrastructural issues.

We developed services domain ontology of >700 concepts obtained from four SaaS application domains: Customer Relationship Management (CRM), Enterprise Resource Planning (ERP), Document Management (DM) and Collaboration. We used established industry classification standards as guiding reference: United Nations Standard Products and Services Code® (UNSPSC®)(The United Nations Standard Products and Services Code® (UNSPSC®). http://www.unspsc.org/.) and North American Industry Classification System (NAICS) (The North American Industry Classification System (NAICS). http://www.census.gov/eos/www/naics/).

The SaaS ontology is represented in the knowledge representation Web Ontology Language (OWL).

Business functions from the SaaS services domain are represented as concepts, (e.g., Project Management and Payroll). Object properties are used for several purposes: describe the service functionalities (supports Business Function property), connect a service to its provider (is ProvidedBy property), describe the service characteristics (has License Type property) and specify the cluster to which the service belongs to (belongsToCluster property). Data properties are used to describe guaranteed QoS values.

The WordNet lexical database (Miller, 1995) is used for semantic expansion of both the service description and the user query. The WordNet superficially resembles a thesaurus in which nouns, verbs and adverbs are grouped into sets of cognitive synonyms (synsets).

OntSaaS discovery subsystem: The SaaS discovery subsystem is responsible for identifying the similarity between the published service capabilities and the functionalities required by the users. Common users usually prefer to submit keyword-based queries in order to describe their requirements. Despite its adequacy to users it may be insufficient to specify the service functionalities. This problem is known as language ambiguity. In more details, the words used by cloud providers in service descriptions are syntactically different than user queries but semantically equivalent which leads to low performance of syntactic matching approaches. The proposed discovery approach aims to reach a balance between enabling appealing keyword-based queries for users and exploiting the advantages of semantic matching. Our aim was achieved by constructing concept-based service requests from the user query after enriching it (semantic expansion using WordNet). Afterward, the service request is matched to semantically-annotated service advertisements via proposed matchmaking algorithm. The proposed business-oriented discovery approach involves minimum user intervention in order to find the SaaS services that match his requirements.

It is worth noting that the proposed discovery approach addresses a major deficiency in existing work on cloud service discovery presented in which cannot help users refine their requirements when there is no feasible solution. Uniquely, we exploit the semantically-rich ontology to expand the user query by broadening its coverage which increases the user chance to find his required service. In more details, if the user query does not return relevant services, a partial matching case, the user engages in another round of interaction in order to expand his original query. In addition to the user interface, the discovery subsystem consists of two main modules: the semantic query processor and the functional matching.

Semantic query processor module: This module is composed of two components: the query preprocessing and the semantic query expansion. The query preprocessing component preprocesses the submitted user query. Semantic Query Expansion (QE) using thesaurus is an example of global methods used in IR systems to increase the recall. In our system, QE is employed in order to avoid the case where the user's vocabulary is different from the SaaS ontology vocabulary. The semantic query expansion component consults the WordNet to retrieve synonyms of the query tokens.

Functional matching module: This module is composed of three components: the concept recommender, the concept-based query matching and the semantic concept expansion. The concept recommender component matches the expanded user query to the service CSVs which results in two cases. The first case is that no matching BFs are found. The user is asked to enter a refined keyword-based query. This case takes place in two conditions. First, when all query tokens are considered stop words and are removed by the query preprocessing component. Second, when the user does not have enough knowledge to make an initial query which is close to ontology concepts.

In the second case, matching BFs are found. Either the matching BFs are directly used without any intervention from the user (automatic) or user-controlled discovery is started which is identified as concept-based query-construction process. The detailed workflow of the proposed user-controlled discovery approach is shown in the activity diagram in Fig. 2.

In this process, the mechanism of the Relevance Feedback (RF) (Salton and McGill, 1986) is adapted to enhance the system performance. In particular, the matching BF concepts are considered the initial result set. The user selects some of the returned concepts. The constructed request consists of the selected BF concepts.

The concept-based query matching component calculates the similarity between the constructed request vector and the service CSVs. We have two cases. Best cluster (s) are identified when a similarity between the CSV and the request vector is above a specified threshold, currently 0.5. The first case is that best cluster (s) are found. We consider clusters with a similarity less than the threshold irrelevant to the user. In this case, the

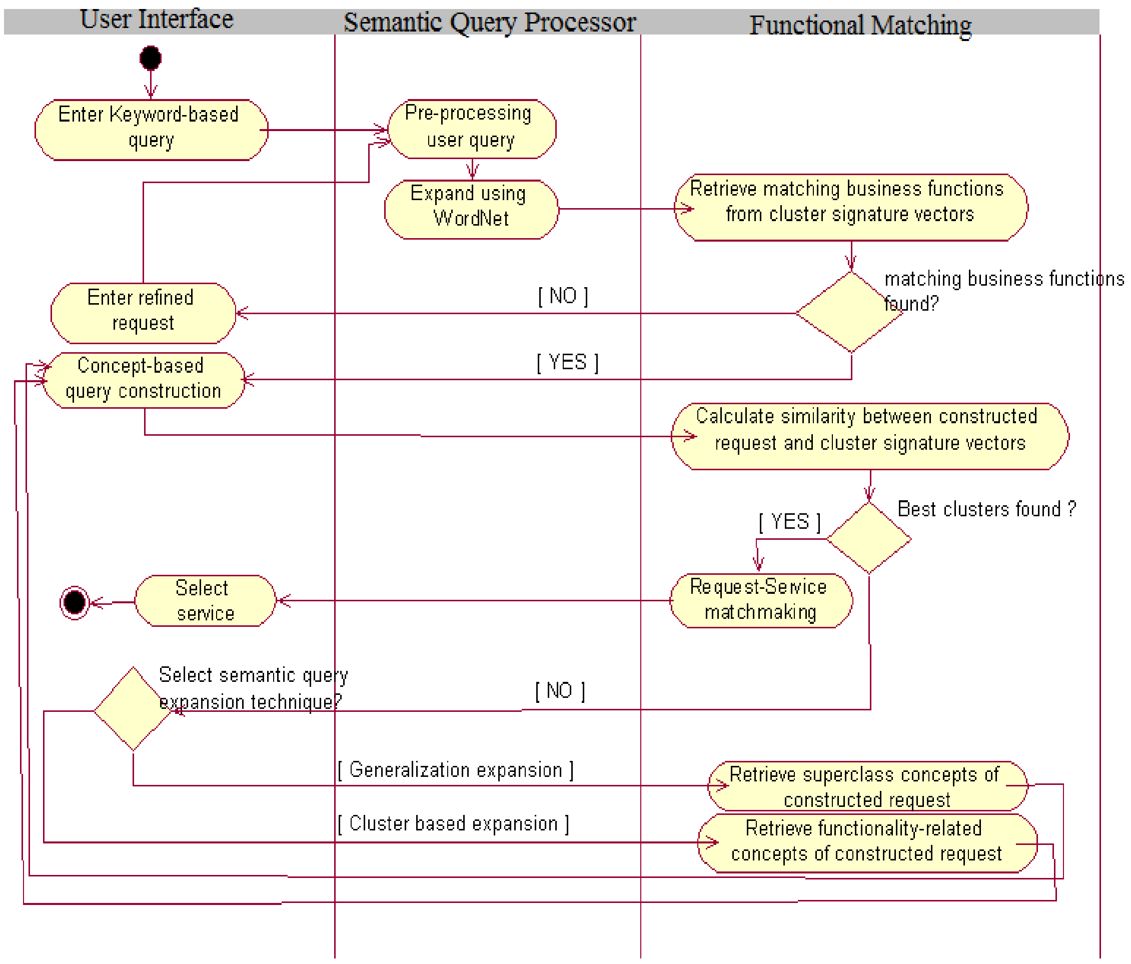
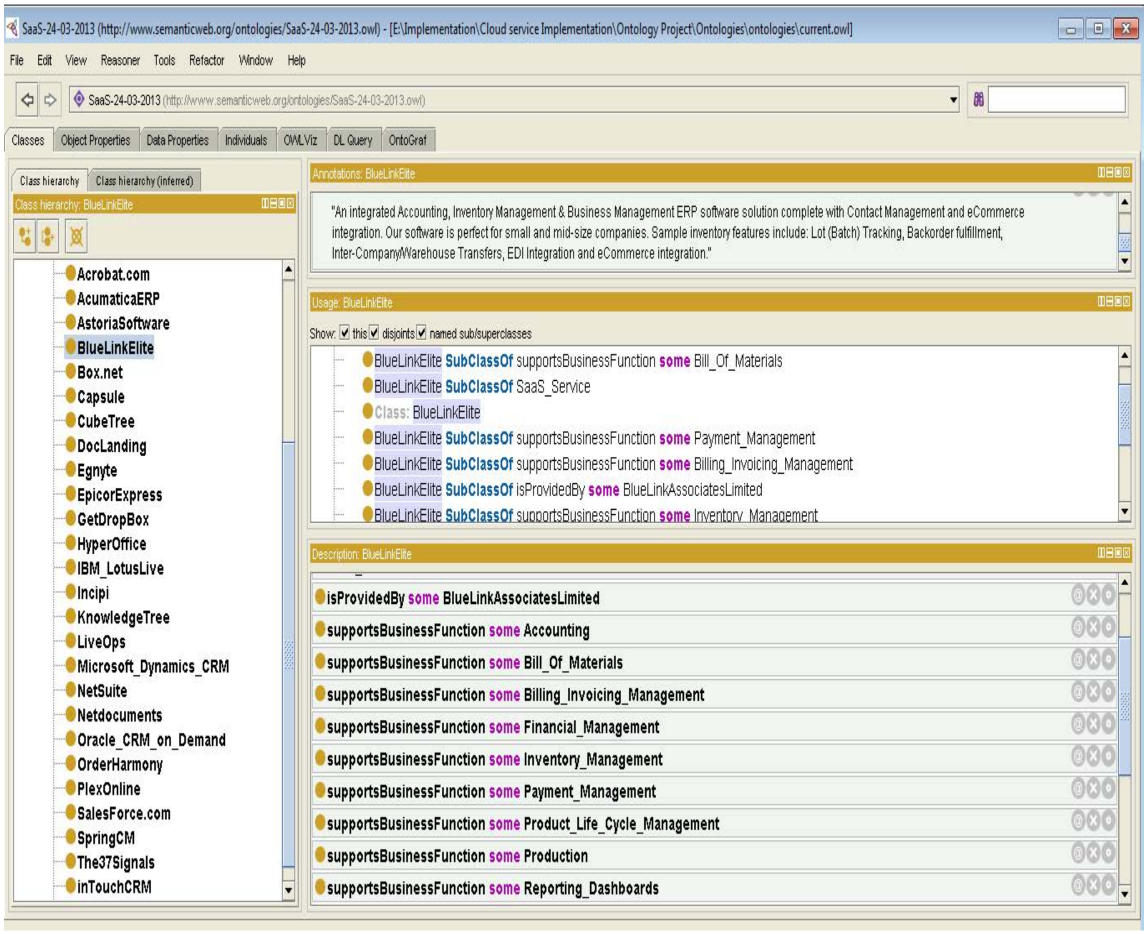

User Interface
Semantic Query Processor
Functional Matching
Enter Keyword-based query
Pre-processing user query
Expand using WordNet
Retrieve matching business functions from cluster signature vectors
matching business functions found?
[NO]
Enter refined request
[YES]
Concept-based query construction
Calculate similarity between constructed request and cluster signature vectors
Best clusters found ?
[YES]
Request-Service matchmaking
Select service
Select semantic query expansion technique?
[NO]
[Generalization expansion]
Retrieve superclass concepts of constructed request
[Cluster based expansion]
Retrieve functionality-related concepts of constructed request

Fig. 2: OntSaaS user-controlled discovery approach: activity diagram

concept-based query matching component compares the constructed request with the service advertisements from the best clusters using the proposed hybrid SaaS request-service matchmaking algorithm. Matching services with a similarity value above threshold, currently 0.5 are displayed to the user and then passed to the selection subsystem.

In the second case, no best cluster (s) are found which means that the selected BFs are highly scattered among service clusters. This case is identified as partial matching process. This means that the similarity of the matching cluster to the user request is below the threshold but above zero. Apparently, we need a refinement for this result. Since these concepts were obtained through an interaction between the user and the system it is assumed that there is some degree of overlap between the returned service concepts and requested service. Therefore, we propose exploiting the semantically-rich ontology environment through the semantic concept expansion component. This is achieved by broadening the request coverage in order to increase the chance of returning relevant services to the user.

The user request expansion uses one of two schemes: generalization expansion and cluster-based expansion. In generalization expansion the semantic concept expansion component generalizes the selected BF concepts to their super class concepts which broadens the retrieval scope of the user request. The super class concepts are displayed to the user. The user chooses some to construct his new request. The concept-based query matching component re-calculates the similarity of the new request vector and the service CSVs. On the other hand, in cluster-based expansion, the semantic concept expansion component retrieves functionally-related BF concepts from the cluster with the maximum similarity. User chooses some to construct his new request. The


SaaS-24-03-2013 (http://www.semanticweb.org/ontologies/SaaS-24-03-2013.owl) - [E:\Implementation\Cloud service Implementation\Ontology Project\Ontologies\ontologies\current.owl]

File Edit View Reasoner Tools Refactor Window Help
SaaS-24-03-2013 (http://www.semanticweb.org/ontologies/SaaS-24-03-2013.owl)
Classes
Object Properties
Data Properties
Individuals
OWLViz
DL Query
OntoGraf
Class hierarchy
Class hierarchy (inferred)
Class hierarchy: BlueLinkElite

Acrobat.com
AcumaticaERP
AstoriaSoftware
BlueLinkElite
Box.net
Capsule
CubeTree
DocLanding
Egnyte
EpicorExpress
GetDropBox
HyperOffice
IBM_LotusLive
Incipi
KnowledgeTree
LiveOps
Microsoft_Dynamics_CRM
NetSuite
Netdocuments
Oracle_CRM_on_Demand
OrderHarmony
PlexOnline
SalesForce.com
SpringCM
The37Signals
inTouchCRM

Annotations: BlueLinkElite

"An integrated Accounting, Inventory Management & Business Management ERP software solution complete with Contact Management and eCommerce integration. Our software is perfect for small and mid-size companies. Sample inventory features include: Lot (Batch) Tracking, Backorder fulfillment, Inter-CompanyWarehouse Transfers, EDI Integration and eCommerce integration."

Usage: BlueLinkElite

Show: ✓ this ✓ disjoints ✓ named sub/superclasses

BlueLinkElite SubClassOf supportsBusinessFunction some Bill_Of_Materials
BlueLinkElite SubClassOf SaaS_Service
Class: BlueLinkElite
BlueLinkElite SubClassOf supportsBusinessFunction some Payment_Management
BlueLinkElite SubClassOf supportsBusinessFunction some Billing_Invoicing_Management
BlueLinkElite SubClassOf isProvidedBy some BlueLinkAssociatesLimited
BlueLinkElite SubClassOf supportsBusinessFunction some Inventory_Management

Description: BlueLinkElite

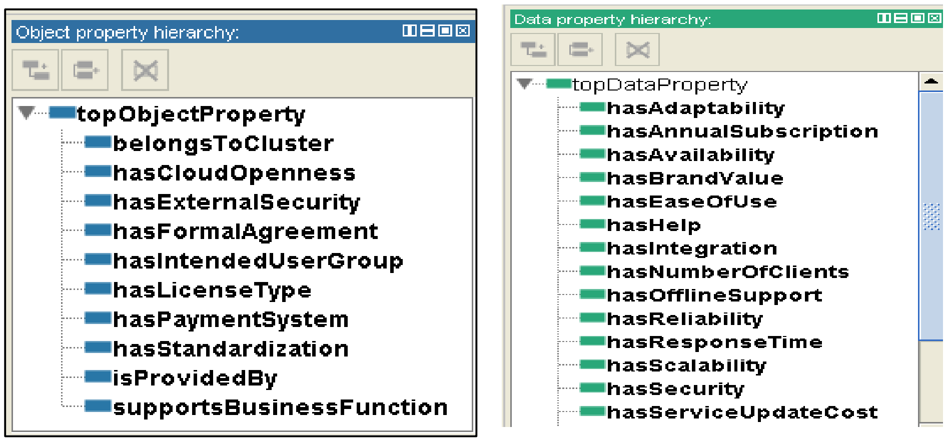
isProvidedBy some BlueLinkAssociatesLimited
supportsBusinessFunction some Accounting
supportsBusinessFunction some Bill_Of_Materials
supportsBusinessFunction some Billing_Invoicing_Management
supportsBusinessFunction some Financial_Management
supportsBusinessFunction some Inventory_Management
supportsBusinessFunction some Payment_Management
supportsBusinessFunction some Product_Life_Cycle_Management
supportsBusinessFunction some Production
supportsBusinessFunction some Reporting_Dashboards

Fig. 3: SaaS Ontology: Modeling of real service offers

concept-based query matching component re-calculates the similarity of new request vector and the service CSVs. Figure 3 presents the pseudo-code of the OntSaaS service discovery Algorithm 1.

Algorithim 1:

```
Algorithm: OntSaaS Service Discovery
Input: User query Q, Discovery approach userControlled
Output: Set of discovered services DS
    BEGIN
    Tokenize Q on delimiters to generate set of querytokens
    Remove stop words to create set of relevantTokens
    Generate expanded query EQ by finding WordNet synonyms of relevantTokens
    Set stems stems of EQ using Stemmer algorithm
    found = match (EQ, CSV, foundClusters)
    IF NOT found THEN
    READ new query Q from user
    GO TO 1
  ELSE
    BF = getBF (foundClusters)
    IF NOT userControlled THEN
        constructedRequest = BF
  ELSE
      DISPLAY BF to the user to start concept-based query construction process
      READ selected business functions constructedRequest
  END IF
  Identify best clusters with similarity above threshold 0.5
   bestClustersFound = sim (constructedRequest, CSV, bestClusters)
   IF bestClustersFound THEN
       FOR each service s in getServices(bestClusters) DO
            simValue = SaaS Request Service Matchmaking (constructed Request, s)
         IF simValue > 0.5 THEN
            DS = DS   Us
         END IF
      END FOR
  ELSE
      READ expansion scheme expansion from user to start partial match process
      CASE expansion OF
         G:  FOR each bf in constructedRequest DO
                 partialBF = partialBF   superClass(bf)
             END FOR
C:  partialBF  =  get  BF  (clusterWithMaxSimilarity  (constructed Request,CSV))
      END CASE
      DISPLAY partialBF to the user
      READ selected business functions constructedRequest
      GO TO 18
  END IF
      END IF
      Return set of discovered services DS
      END
```



Object property hierarchy:

- topObjectProperty
  - belongsToCluster
  - hasCloudOpenness
  - hasExternalSecurity
  - hasFormalAgreement
  - hasIntendedUserGroup
  - hasLicenseType
  - hasPaymentSystem
  - hasStandardization
  - isProvidedBy
  - supportsBusinessFunction

Data property hierarchy:

- topDataProperty
  - hasAdaptability
  - hasAnnualSubscription
  - hasAvailability
  - hasBrandValue
  - hasEaseOfUse
  - hasHelp
  - hasIntegration
  - hasNumberOfClients
  - hasOfflineSupport
  - hasReliability
  - hasResponseTime
  - hasScalability
  - hasSecurity
  - hasServiceUpdateCost

Fig. 4: SaaS Ontology: a) Object properties view and and b) Data properties view

Algorithm 2:

```
SaaS Request-Service Matchmaking
Input: Constructed Request R and service S
Output: Overall similarity: sim(R, S)
BEGIN
Calculate the features similarity
```

$$\text{sim}_f(R,S)=\frac{(?\cup bj)_R\cap ubj_S|\quad)}{|obj_R\quad|}$$

```
    IF sim_f(R, S)<  THEN
        Calculate the hierarchical similarity $sim_{ontH}$ (R, S)
        sum = 0
        FOR each concept ci in R where 1<= i <= n DO
           maxSim = -1
           FOR each concept cj in S where 1<= j <= m DO
              IF ci is a child of cj THEN
                 $Sim_{ontH}(c_i, c_j) = 1$
              ELSE IF ci is a parent of  cj THEN
                 $Sim_{ontH}(ci, cj) = 1-0.5$
              ELSE IF ci is a descendent of  cj THEN
                 $Sim_{ontH}(ci, cj) = 0.75$
              ELSE IF ci is an ascendant of  cj THEN
                 $Sim_{ontH}(ci, cj) = 0.5$
              ELSE IF ci and  cj are siblings THEN
                 $Sim_{ontH}(ci, cj) = simLIN(ci, cj$
              ELSE
           $Sim_{ontH}(ci, cj) = 0$
              END IF
           If $Sim_{ontH}(ci, cj)$ =>MaxSim
                maxSim =  $Sim_{ontH}(ci, cj)$
           END IF
        END FOR
       sum = sum + maxSim
       END FOR
       $Sim_{ontH}(R, S\ ) = sum / n$
       Calculate
    sim (R, S) = a. $sum_f(R, S)$+b. $Sum_{ontH}(R, S)$
    ELSE
     sim(R, S) = 1
    END IF
```

Request-service matchmaking algorithm: In this section, we propose a hybrid SaaS request-service matchmaking algorithm which is used by the concept-based query matching component to compare the constructed request with the service advertisements from the best clusters. The proposed request-service matchmaking algorithm is an enhanced version of our SaaS services matchmaking algorithm (Afify *et al*., 2013, 2014a, b). The objective of the proposed request-service matchmaking algorithm is to find services that fulfill all or part of the required functionality requested by the user. It utilizes ontology-based matching which makes use of concept hierarchical structure. The ontological hierarchical structure considers both the distance and content similarity models (Cross *et al*., 2013; Dong *et al*., 2011a). Figure 4 presents the pseudo-code of the proposed SaaS request-service matchmaking algorithm.

First, the feature similarity is calculated which accounts for the mutual BFs between the user request and the service. Second, the hierarchical similarity is calculated which takes into consideration any ontological relationship among the request and the service unique BFs. The request-service hierarchical similarity is calculated by computing the average maximum inter-similarity between the unique concepts. First, each concept from the request R is compared with all concepts of the service S and the maximum similarity value is taken and then repeats for all R concepts. Second, the average of the n comparisons is calculated where n represents the number of unique BFs in the user request R. To compare two concepts we have six cases. Table 1 presents the concept-concept h ierarchical similarity cases with detailed justification. The overall similarity between user request and a service is calculated by taking weighted average of features and hierarchical similarity measures using Eq. 1:

Table 1: Request-service concepts hierarchical similarity value derivation

Case	Similarity value	Hierarchical similarity calculation justification
Request concept is parent child of service concept	1	Since SaaS domain ontology concepts are linked with is-a relations, then the requested function is supported by the service. The similarity is maximal
Request concept is of service concept	$1\text{-}0.5^{\sigma Rdopth}$	The service supports one sub-type/category of the requested function. Our rationale parent is to use the request concept depth to account forthe closeness between parent-child concepts. Concepts at upper levels are more abstract while concepts at lower levels are more specific. Consequently, parent-child concepts at lower levels of ontology are more related than at upper levels
Request concept is descendent of service concept	$0.75^{level(\alpha R.\alpha S)}$	The service is a broad type of the requested function. The rationale is that the number of levels between the two concepts account for the closeness between them
Request concept is ascendant of service concept	$0.5^{level(cR.cS)}$	The service is a specialization of the requested function. It supports only a minor part of the requested function. rationale is that the number of levels between the two concepts account for the closeness between them. This case results in similarity values lower than that of case #3
Request concept and service is concept are siblings	$sim_{LIN}(c_R.c_S)$	Two concepts share the same parent. Then distance similarity is irrelevant. In this case, the content-based similarity adopted to account for the amount of shared information between the two concepts. We use LIN similarity measure
Two concepts meet at root node	0	Two concepts do not subsume each other in any way

$$\text{sim}(R,S) = \alpha.\text{sim}_f(R,S) + b.\text{sim}_{ontH}(R,S) \qquad (1)$$

where, a and b are weights that reflect significance of each similarity measure. These weights are assigned equal value of 0.5.

RESULTS AND DISCUSSION

Case studies and experimental evaluation: To demonstrate the effectiveness of the proposed system, we implemented a prototype with real and synthetic cloud data. Experiments were conducted on an Intel Core i3 2.13 GHz processor, 5.0 GB RAM running under Windows 7 Ultimate. The system was built using Java, Jena API and WordNet API incorporated in Eclipse IDE.

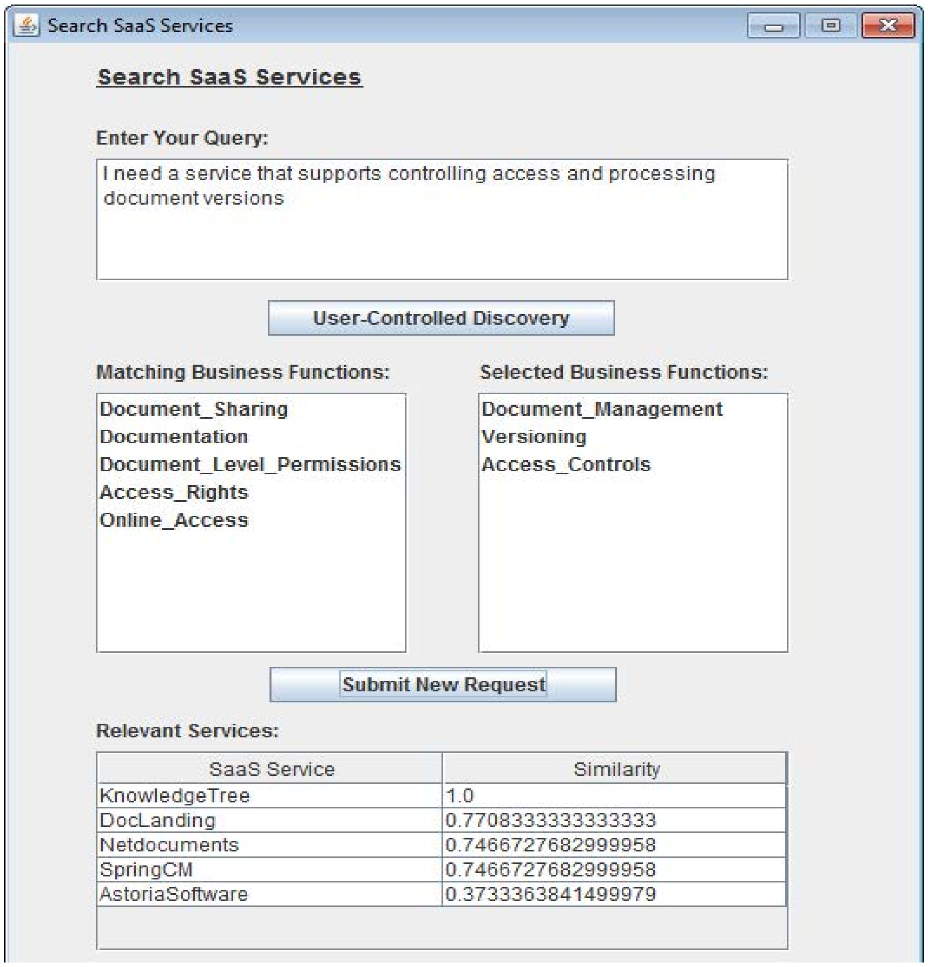
We built a data set of 500 SaaS service offers. Specifically, 40 services are live services and the remaining are pseudo services generated by adapting the real services with some changes. Real cloud SaaS offers were collected from the cloud provider portals. In the following subsections, we present the SaaS ontology implementation, case study scenarios and the experimental evaluation.

SaaS ontology implementation: This section describes the implementation of the SaaS ontology. The SaaS ontology was implemented using Protege 4.1 ontology editor (Horridge *et al*., 2004). Modeling of real service offers is demonstrated by example in Fig. 3. The blue link elite service description is stored in the annotations section. The service provider blue link associates limited is presented using the object property is provided by. The service features are mapped into BFs using the object property supports business function. As shown in Fig. 5, the blue link elite service supports accounting, payment management and inventory management among others. The unified SaaS ontology object properties are shown in Fig. 4a. Some of the data properties are shown in Fig. 4b. More details on the developed ontology are provided in our previous work (Afify *et al*., 2014a).

Case study scenarios: We present three service discovery scenarios. The objective of the first discovery scenario is to compare the proposed concept-based query construction discovery process against the service discovery using standard Cosine Similarity measure with TF/IDF weighting model (Salton and McGill, 1986). The objective of the second discovery scenario is to compare the automated vs. user-controlled discovery approaches. The objective of the third discovery scenario is to evaluate the effectiveness of the partial matching case processing.

In general, in order to search for a service with specific functionalities, the user enters a keyword-based query with the required features. We present three case study scenarios for the discovery process. The first scenario describes a successful discovery that uses the concept-based query construction approach against Cosine-based discovery while the second scenario compares the automated vs. user-controlled discovery processes. Finally, the third scenario presents a partial matching case that uses the generalization expansion approach.

User-controlled vs Cosine-based discovery scenario: The objective of this scenario is to demonstrate the efficiency of the proposed user-controlled discovery approach against the Cosine-based discovery approach. In general, to discover SaaS services, the user enters his key word-based query using his own terminology.



I need a service that supports controlling access and processing document versions

Matching Business Functions:	Selected Business Functions:
Document_Sharing	Document_Management
Documentation	Versioning
Document_Level_Permissions	Access_Controls
Access_Rights	
Online_Access	

SaaS Service	Similarity
KnowledgeTree	1.0
DocLanding	0.7708333333333333
Netdocuments	0.7466727682999958
SpringCM	0.7466727682999958
AstoriaSoftware	0.373363841499979

Fig. 5: SaaS Services user-controlled discovery (concept-based query construction)

In the Cosine-based approach, the semantic similarity between the user query and the service CSVs was calculated using the Cosine similarity. Services that belong to the cluster with the best similarity were retrieved. To rank the discovered services, the semantic similarity between the user query and these service descriptions was calculated using Cosine similarity. For example, using the search query "I need a service that supports user access and document versions", the relevant services identified by the system were NetDocuments with a similarity of 0.22, DocLanding with a similarity of 0.18, KnowledgeTree with a similarity of 0.12, AstoriaSoftware with a similarity of 0.07 and SpringCM with a similarity of 0.05. Notably, the resulting service similarities to the user query are generally low which misleads the user to consider these services are irrelevant to him. The reason is that Cosine-based discovery matches user query to service description only which may not contain complete or accurate details about service functionalities.

On the other hand, in this study, the matching is business-oriented it relies on ontology-based matchmaking of the service features. Figure 5 demonstrates the use of the user-controlled discovery of the same query. The system matches the query key terms to the service CSVs and displays the matching BF concepts in the matching business functions list. The concept recommendation process provides an excellent chance for the user to improve his query by selecting the concepts that exactly match his required functionalities (e.g., document-management, versioning and access-controls). The selected BFs are moved to the selected business functions list. Finally, the user presses the submit new request button. The new request is matched against the service CSVs. Finally, the relevant services are ranked based on their similarity to the user request.

As shown in Fig. 5, the best service that fulfills the user request is the knowledge tree service with a similarity of 1. This result means that the knowledge tree service supports all the BFs requested by the user. While the other services support some/none of the requested BFs. The similarity values returned by the request-service matchmaking algorithm represent the combination of features and hierarchical similarities. These results reveal the actual relevance of the services to the user query as opposed to the Cosine-based approach.

Automated vs. User-controlled discovery scenario: The objective of this scenario is to compare between the automated and user-controlled discovery approaches. Using the following search query "I need a service that handles translating and searching in documents". In case of automated discovery, the system returned the following BFs: Document-management, translation, document-sharing, documentation, search, search-OCR, search-full-text, document-level-permissions, search metadata.

All matching BFs were used in the discovery process. Relevant services returned by the system were net documents with a similarity of 0.42 and knowldege tree with a similarity of 0.33. Matching a large number of BFs which may not be functionally-related, resulted in missing some relevant services.

The user selected the following BFs document sharing, search and translation. Relevant services returned by the system were astoria software with a similarity of 0.59, net documents with a similarity of 0.59, spring CM with a similarity of 0.51, knowldege tree with a similarity of 0.39 and doc landing with a similarity of 0.31. The system retrieved the services that best match user functionality requirements, against a recall value of 0.4 using automated discovery approach.

In another scenario, the user selected the following BFs document-management, search and translation. Relevant services returned were astoria software with a similarity of 0.67, knowldege tree with a similarity of 0.48 and net documents with a similarity of 0.33. The recall value is 0.66 for this scenario.

In summary, the automated discovery approach takes less time but suffers from low recall values. On the other hand, the user-controlled concept-based query construction discovery approach takes more time (taken by the user to select the BFs) but achieves higher recall values with average improvement 100%. Both achieve high precision values.

Enhancing discovery recall

Partial matching scenario: The objective of the partial matching scenario is to demonstrate the case in which no best clusters were found after matching the constructed request to the service CSVs. Using the following search query "I need a service that supports recruiting employees, payroll reports and accounts". The concept recommender component returned the following BFs: Accounts-Payable-Recievable, Inventory-Reports, Audit-Reporting, Payroll-Management, Recruitment, Reporting-Dashboards, Reporting, Financial-Operational Reporting and Accounting. The user selected Accounts Payable-Recievable, Payroll-Management and Recruitment to construct a new request. No relevant services were identified by the system.

In order to support the user to find suitable services, the semantic concept expansion component provides two expansion approaches. This scenario demonstrates the use of the generalization expansion approach. Super class concepts of the concept-based constructed request are displayed to the user which are Accounts, Payroll and Human-Resource respectively. The new request is matched to the service CSVs. Finally, the relevant services returned by the system were PlexOnline with a similarity of 0.29, NetSuite with a similarity of 0.25, Blue Link Elite with a similarity of 0.22, AcumaticaERP with a similarity of 0.19, EpicorExpress with a similarity of 0.17 and Order Harmony with a similarity of 0.13.

Experimental evaluation

Experiment 1: The objective of this experiment is threefold: to calculate the time taken by the registration system to semantically annotate the service description, to study the effect of the semantic expansion of the service description using WordNet on the annotation process and to compute the precision and recall of the semantic annotations by the concept recommender component.

To achieve the first objective, we computed the processing time taken by the concept recommender component to semantically annotate the service description which has been expanded using WordNet. As shown in Fig. 6, the time taken by the semantic annotation of service description during the service registration is negligible.

In order to achieve the second objective, we have analyzed the semantic annotation process in two cases, with and without using Word Net. We compare the number of matching BFs returned in the two cases. Results in Fig. 7 show that the semantic expansion of the service description generally increases the number of retrieved BFs. However, this increase shows a discrepancy and is not proportional to the number of expanded description terms. It is greatly dependent on the terms used by the cloud provider in describing his service and how close they are to the concepts in the services domain ontology.

Table 2: Precision and recall of semantic annotations

	Precision			Recall		
Original description terms	Using original service description	Using expanded service description	Change in precision (%)	Using original service description	Using expanded service description	Change in recall (%)
50	1	0.86	-14	0.92	1	8.6
101	1	0.7	-30	0.88	1	13.6
152	1	0.81	-19	1	0.93	-7
203	1	0.83	-17	0.95	1	5.2
257	1	0.76	-24	0.9	1	11

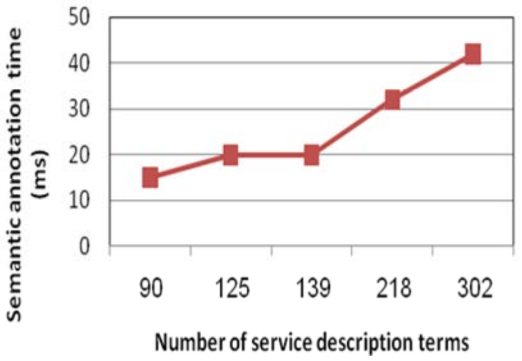


Fig. 6: Semantic annotation time of service descriptions

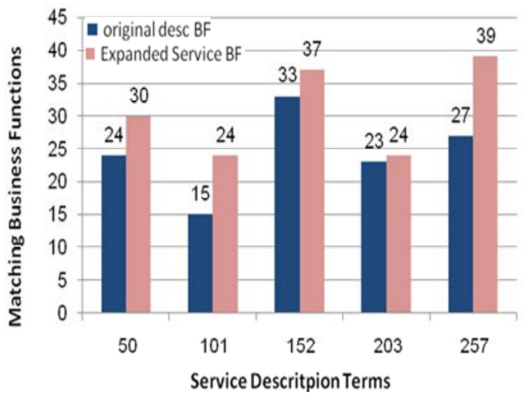


Fig. 7: Service description semantic annotation process

In order to achieve the third objective, we computed the precision and recall of the semantic annotations returned by the concept recommender component during the service registration in two cases, with and without using WordNet. The precision is calculated using Eq. 2, while the recall is calculated using Eq. 3. Table 2 shows the precision and recall values of the semantic annotations of services with different number of description terms.

$$\text{Precision} = \frac{\text{Number of correct returned business functions}}{\text{Number of returned business functions}} \quad (2)$$

$$\text{Recall} = \frac{\text{Number of correct returned business functions}}{\text{Number of expected business functions}} \quad (3)$$

It is apparent that the semantic expansion of the service description has a negative effect on the precision of the semantic annotation process by an average deterioration of 20%. On the other hand it has a generally positive effect on the recall of semantic annotation process.

Experiment 2: The objective of this experiment is to evaluate the performance of the automated concept-based query construction approach in respect of three evaluation metrics: time, service utility and success rate. We compare the proposed discovery approach to the standard Cosine-based discovery employed in a number of service discovery proposals (Garcia *et al*., 2014; Ding *et al*., 2010; Platzer and Dustdar, 2005; Paliwal *et al*., 2012; Hao *et al*., 2010) in order to compare the service query to service descriptions.

First, the objective is to evaluate the time taken to discover relevant services for a user query. We used different clustering threshold values from 0.1-0.9 for user queries that consists of 5-10 key terms. In case of the automated discovery approach, the processing time is composed of time taken by the concept recommender component to find matching BFs for user query and the time taken by the concept-based query matching component to match the new request to the service CSVs. From Fig. 8, we can conclude that there is no overhead introduced by the proposed automated discovery approach in respect of the processing time.

Second, the objective is to compare between the Cosine-based discovery approach and the proposed automated concept-based query construction approach in respect of evaluation metrics service utility and success rate. The service utility is the similarity between the user query and service description which is calculated by the SaaS request-service matchmaking algorithm in the proposed discovery process where its value range is 0-1. The success rate is calculated by the number of

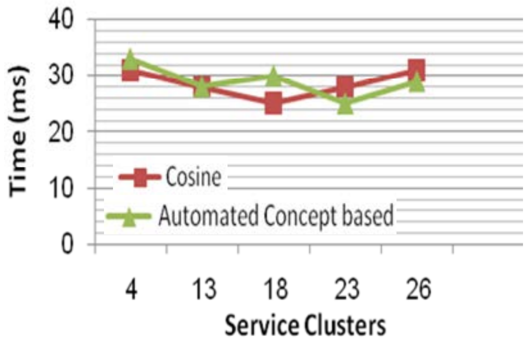


Fig. 8: Service discovery process time of cosine vs automated concept-based query construction

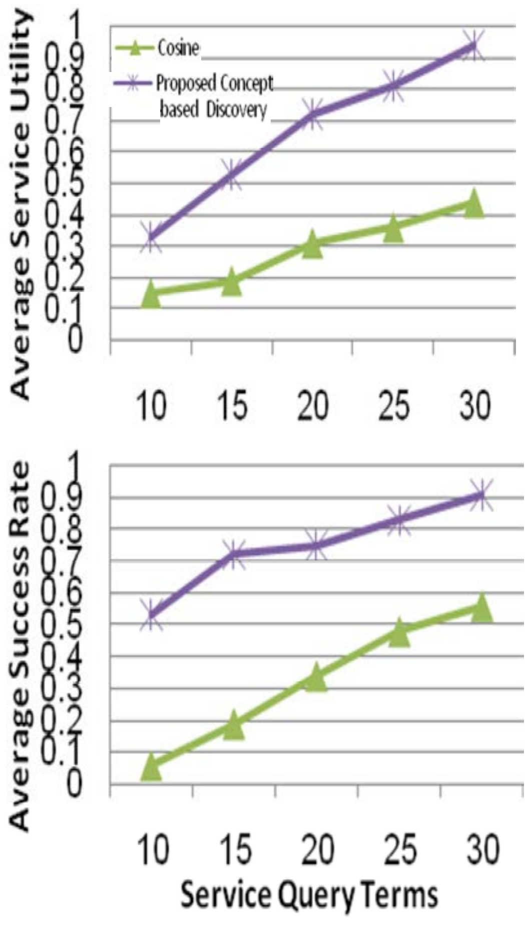


Fig. 9: Service discovery process: a) service utility; b) success rate

successes/the number of attempts. It is assumed that an attempt fails if the service utility is less than specified threshold 0.3 where its value range is 0-1.

For our experiments, 5 queries were created for each of the four application domains included in the SaaS ontology. For each domain, we varied the number of service query terms in range from 10-30 sing step of 5. The average values of the service utilities and success rates were reported over the four executions. The results are shown in Fig. 9 As shown in Fig. 9, the number of service query terms has the same effect on the average service utility and success rate of both approaches. This behavior is expected since increasing the number of service query terms usually increases the probability of matching between query terms and the service description. Consequently, the service utility and success rate increases. The results show that the proposed concept-based discovery outstandingly outperforms the Cosine-based discovery approach along all number of service query terms. This performance enhancement is due to the unique process of transforming the user keyword-based query into business functions and then matching them to the services functional metadata using the request-service matchmaking algorithm.

The experimental results show that the user controlled discovery outperforms the standard Cosine based discovery in respect of the service utility (similarity relevance) and success rate with comparable execution time. These results are due to the unique ontology-based matchmaking of the semantically-constructed enriched user query and semantically-annotated service advertisements. The ontology-based matchmaking enabled the identification of semantic relationships between different concepts in the user query and service advertisements as opposed to merely exact matching of the concepts in the Cosine-based discovery. This performance enhancement in the similarity relevance contributes positively to the user acceptance of the discovered services. Moreover, displaying the similarity values justifies the rationalization of the results which improves the credibility of the discovery system.

CONCLUSION

The incompatibility and lack of standardization of the cloud services publication represent the key factors that hinder the broad adoption of cloud computing. Motivated by these findings, in this study, we proposed OntSaaS, a semantic-based cloud service publication and discovery system for SaaS services.

The OntSaaS service publication process supports standardized representation for SaaS services offered by different cloud providers in a single semantic-based service catalogue. The catalogue integrates service functionality and quality information. Utilizing semantic annotation and concept recommendation, a guided registration process was proposed to assist the cloud provider to map the service features to service domain ontology concepts.

The OntSaaS service discovery process provides efficient user-controlled discovery capabilities for service users. Semantically-annotated service advertisements are matched to the semantically-constructed service requests by users. Query expansion and relevance feedback approaches are adopted in order to improve the efficiency of the discovery process. Prototypical evaluation of the system proved its performance enhancement in respect of the service utility and success rate. Results showed that the concept recommendation approach employed decreased the service registration process time. Moreover, the proposed matchmaking algorithm similarity results revealed the actual relevance of the offered services to the user requests. Finally, the proposed ontology-based expansion approach for the user request improved the user opportunity to find appropriate services to his requirements in case of discovery partial match.

REFERENCES

Afify, Y.M., I.F. Moawad, N.L. Badr and M.F. Tolba, 2013. A semantic-based Software-as-a-Service (saas) discovery and selection system. Proceedings of the 2013 8th International Conference on Computer Engineering and Systems (ICCES), November 26-28, 2013, IEEE, New York, USA., ISBN: 978-1-4799-0080-0, pp: 57-63.

Afify, Y.M., I.F. Moawad, N.L. Badr and M.F. Tolba, 2014a. Concept Recommendation System for Cloud Services Advertisement. In: Advanced Machine Learning Technologies and Applications. Aboul, E.H., M.F. Tolba and A.T. Azar (Eds.). Springer International Publishing, New York, USA., ISBN: 978-3-319-13460-4, pp: 57-66.

Afify, Y.M., I.F. Moawad, N.L. Badr and M.F. Tolba, 2014b. Cloud Services Discovery and Selection: Survey and New Semantic-Based System. In: Bio-Inspiring Cyber Security and Cloud Services: Trends and Innovations. Hassanien, A.E., T.H. Kim, J. Kacprzyk and A.A. Ismail (Eds.). Springer Berlin Heidelberg, Berlin, Germany, ISBN: 978-3-662-43615-8, pp: 449-477.

Bernstein, D. and D. Vij, 2010. Using semantic web ontology for intercloud directories and exchanges. Proceedings of the International Conference on Internet Computing, July 12-15, 2010, Icomp'10 Publisher, Las Vegas, Nevada, pp: 18-24.

Born, M., A. Filipowska, M. Kaczmarek, I. Markovic and M. Starzecka et al., 2008. Business functions ontology and its application in semantic business process modelling. Proceedings of the 19th Australasian Conference on Information Systems (ACIS), December 3-5, 2008, AIS Publications, Christchurch, Australia, pp: 136-145.

Chang, Y.S., C.T. Yang and Y.C. Luo, 2011. An ontology based agent generation for information retrieval on cloud environment. J. Univers. Comput. Sci., 17: 1135-1160.

Chen, F., X. Bai and B. Liu, 2011. Efficient Service Discovery for Cloud Computing Environments. In: Advanced Research on Computer Science and Information Engineering. Gang, S. and H. Xiong (Eds.). Springer Berlin Heidelberg, Berlin, Germany, ISBN: 978-3-642-21410-3, pp: 443-448.

Cross, V., X. Yu and X. Hu, 2013. Unifying ontological similarity measures: A theoretical and empirical investigation. Int. J. Approximate Reasoning, 54: 861-875.

Dastjerdi, A., S. Tabatabaei and R. Buyya, 2010. An effective architecture for automated appliance management system applying ontology-based cloud discovery. Proceedins of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, May 17-20, 2010, Melbourne, Australia, pp: 104-112.

Ding, D., L. Liu and H. Schmeck, 2010. Service discovery in self-organizing service-oriented environments. Proceedings of the 2010 IEEE Conrference on Asia-Pacific Services Computing Conference (APSCC), December 6-10, 2010, IEEE, New York, USA., ISBN: 978-1-4244-9396-8, pp: 717-724.

Dong, H., F.K. Hussain and E. Chang, 2011a. A context-aware semantic similarity model for ontology environments. Concurrency Comput. Pract. Experience, 23: 505-524.

Dong, H., F.K. Hussain and E. Chang, 2011b. A service concept recommendation system for enhancing the dependability of semantic service matchmakers in the service ecosystem environment. J. Network Comput. Appl., 34: 619-631.

Dong, H., F.K. Hussain and E. Chang, 2011c. A service search engine for the industrial digital ecosystems. IEEE. Trans. Ind. Electron., 58: 2183-2196.

Dong, H., F.K. Hussain and E. Chang, 2013. UCOSAIS: A Framework for User-Centered Online Service Advertising Information Search. In: Web Information Systems Engineering (WISE) 2013. Xuemin, L., M. Yannis, S. Divesh and H. Guangyan (Eds.). Springer Berlin Heidelberg, Berlin, Germany, ISBN: 978-3-642-41229-5, pp: 267-276.

Fortis, T.F., V.I. Munteanu and V. Negru, 2012. Towards an ontology for cloud services. Proceedings of the 2012 6th International Conference on Complex Intelligent and Software Intensive Systems (CISIS), July 4-6, 2012, IEEE, New York, USA., ISBN: 978-1-4673-1233-2, pp: 787-792.

Garcia, M.A.R., R.V. Garcia, F.G. Sanchez and J.J.S. Zapater, 2014. Ontology-based annotation and retrieval of services in the cloud. Knowl. Based Syst., 56: 15-25.

Garg, S.K., S. Versteeg and R. Buyya, 2013. A framework for ranking of cloud computing services. Future Generation Comput. Syst., 29: 1012-1023.

Hao, Y., Y. Zhang and J. Cao, 2010. Web services discovery and rank: An information retrieval approach. Future Generation Comput. Syst., 26: 1053-1062.

Hepp, M., 2006. Products and services ontologies: A methodology for deriving OWL ontologies from industrial categorization standards. Int. J. Semant. Web Inf. Syst. (IJSWIS.), 2: 72-99.

Hofer, C.N. and G. Karagiannis, 2011. Cloud computing services: Taxonomy and comparison. J. Internet Serv. Appl., 2: 81-94.

Horridge, M., H. Knublauch, A. Rector, R. Stevens and C. Wroe, 2004. A Practical Guide to Building OWL Ontologies Using the Protege-OWL Plugin and CO-ODE Tools. 1st Edn., University of Manchester, Manchester, England, Pages: 118.

Joshi, K.P., Y. Yesha and T. Finin, 2014. Automating cloud services life cycle through semantic technologies. IEEE. Trans. Serv. Comput., 7: 109-122.

Kang, J. and K.M. Sim, 2016. Ontology-enhanced agent-based cloud service discovery. Int. J. Cloud Comput., 5: 144-171.

Lei, Y., Z. Wang, L. Meng and X. Qiu, 2014. Clustering and recommendation for semantic web service in time series. Trans. Internet Inf. Syst., 8: 2743-7362.

Limam, N. and R. Boutaba, 2010. Assessing software service quality and trustworthiness at selection time. Trans. Software Engin., 36: 559-574.

Lin, W., W. Dou, Z. Xu and J. Chen, 2013. A QoS-aware service discovery method for elastic cloud computing in an unstructured peer-to-peer network. Concurrency Comput. Pract. Exp., 25: 1843-1860.

Miller, G.A., 1995. WordNet: A lexical database for English. Commun. ACM, 38: 39-41.

Mindruta, C. and T.F. Fortis, 2013. A semantic registry for cloud services. Proceedings of the 2013 27th International Conference on Advanced Information Networking and Applications Workshops (WAINA), March 25-28, 2013, IEEE, New York, USA., ISBN: 978-1-4673-6239-9, pp: 1247-1252.

Modica, G. and O. Tomarchio, 2015. Matching the business perspectives of providers and customers in future cloud markets. Cluster Comput., 18: 457-475.

Moscato, F., R. Aversa, B. Martino, T.F. Fortis and V. Munteanu, 2011. An analysis of mosaic ontology for cloud resources annotation. Proceedings of the 2011 Federated Conference on Computer Science and Information Systems (FedCSIS), September 18-21, 2011, IEEE, New York, USA., ISBN: 978-1-4577-0041-5, pp: 973-980.

Nabeeh, N.A., H.A. Ghareeb and A.M. Riad, 2015. Integrating software agents and web services in service oriented architecture based cloud services discovery framework. J. Convergence Inf. Technol., 10: 67-79.

Noor, T.H., Q.Z. Sheng, A. Alfazi, A.H. Ngu and J. Law, 2013. CSCE: A crawler engine for cloud services discovery on the world wide web. Proceedings of the 2013 IEEE 20th International Conference on Web Services (ICWS), June 28-July 3, 2013, IEEE, New York, USA., ISBN: 978-0-7695-5025-1, pp: 443-450.

Paliwal, A.V., B. Shafiq, J. Vaidya, H. Xiong and N. Adam, 2012. Semantics-based automated service discovery. Serv. Computing IEEE. Trans., 5: 260-275.

Platzer, C. and S. Dustdar, 2005. A vector space search engine for web services. Proceedings of the 3rd European Conference on Web Services (ECOWS'05), November 14-16, 2005, IEEE, New York, USA., ISBN: 0-7695-2484-2, pp: 1-9.

Reshmy, K.R. and S.K. Srivatsa, 2005. Automatic ontology generation for semantic search system using data mining techniques. Asia J. Inform. Technol., 4: 1187-1194.

Salton, G. and M.J. McGill, 1986. Introduction to Modern Information Retrieval. McGraw-Hill Companies, Pennsylvania, USA.,.

Sim, K.M., 2012. Agent-based cloud computing. IEEE Trans. Services Comput., 5: 564-577.

Tahamtan, A., S.A. Beheshti, A. Anjomshoaa and A.M. Tjoa, 2012. A cloud repository and discovery framework based on a unified business and cloud service ontology. Proceedings of the 2012 IEEE 8th Conference on World Congress on Services, June 24-29, 2012, IEEE, New York, USA., ISBN: 978-1-4673-3053-4, pp: 203-210.

Tserpes, K., F. Aisopos, D. Kyriazis and T. Varvarigou, 2012. A recommender mechanism for service selection in service-oriented environments. Future Generation Comput. Syst., 28: 1285-1294.

Youseff, L., M. Butrico and D. DaSilva, 2008. Toward a unified ontology of cloud computing. Proceedings of the 2008 Conference on Grid Computing Environments Workshop, November 12-16, 2008, IEEE, New York, USA., ISBN: 978-1-4244-2860-1, pp: 1-10.

Yu, Q., 2015. Cloud Rec: A framework for personalized service recommendation in the cloud. Knowl. Inf. Syst., 43: 417-443.

Zhao, L., Y. Ren, M. Li and K. Sakurai, 2012. Flexible service selection with user-specific QoS support in service-oriented architecture. J. Network Comput. Appl., 35: 962-973.