IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received August 19, 2019, accepted September 1, 2019, date of publication September 5, 2019,
date of current version September 20, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2939543

A Centralised Cloud Services Repository (CCSR) Framework for Optimal Cloud Service Advertisement Discovery From Heterogenous Web Portals

ASMA MUSABAH ALKALBANI[1], WALAYAT HUSSAIN[2], AND JUNG YOON KIM[3]
[1]Information Technology Department, College of Applied Sciences, Ibri 511, Oman
[2]Faculty of Engineering and Information Technology, University of Technology Sydney, Sydney, NSW 2007, Australia
[3]Graduate School of Game, Gachon University, Seongnam 13120, South Korea

Corresponding authors: Asma Musabah Alkalbani (asmam.ibr@cas.edu.om) and Jung Yoon Kim (kjyoon79@gmail.com)

ABSTRACT A cloud service marketplace is the first point for a consumer to discovery, select and possible composition of different services. Although there are some private cloud service marketplaces, such as Microsoft Azure, that allow consumers to search service advertainment belonging to a given vendor. However, due to an increase in the number of cloud service advertisement, a consumer needs to find related services across the worldwide web (WWW). A consumer mostly uses a search engine such as Google, Bing, for the service advertisement discovery. However, these search engines are insufficient in retrieving related cloud services advertainments on time. There is a need for a framework that effectively and efficiently discovery of the related service advertisement for ordinary users. This paper addresses the issue by proposing a user-friendly harvester and a centralised cloud service repository framework. The proposed Centralised Cloud Service Repository (CCSR) framework has two modules - Harvesting as-a-Service (HaaS) and the service repository module. The HaaS module allows users to extract real-time data from the web and make it available to different file format without the need to write any code. The service repository module provides a centralised cloud service repository that enables a consumer for efficient and effective cloud service discovery. We validate and demonstrate the suitability of our framework by comparing its efficiency and feasibility with three widely used open-source harvesters. From the evaluative result, we observe that when we harvest a large number of services advertisements, the HaaS is more efficient compared with the traditional harvesting tools. Our cloud services advertisements dataset is publicly available for future research at: http://cloudmarketregistry.com/cloud-market-registry/home.html.

INDEX TERMS Cloud services discovery, web harvesting, service advertisements, ontology, centralized repository, heterogeneous data.

I. INTRODUCTION

The cloud computing paradigm is a new model of delivering computing resources, such as online applications, storage and networks, as a service over the World Wide Web (WWW). It focuses on sharing IT resources over a scalable network called the *cloud* [1]. Cloud computing is a multi-domain environment that offers thousands of online services, which makes the discovery of cloud services a complex and multifaceted task. Given the fact that end-users' requirements vary and that cloud service providers provide a range of cloud services with only slight variations, cloud service selection is a complex yet vital problem to address [2]. The cloud offers three types of service models: *infrastructure as a service* (IaaS), *platform as a service* (PaaS), and *software as a service* (SaaS). The cloud provider offers these services through cloud service advertising. The term *'cloud service advertising'* refers to a cloud service description that present via media, which is an essential factor in any marketplace [3], [4]. The cloud service description describes a complete service

The associate editor coordinating the review of this manuscript and approving it for publication was Honghao Gao.

offered by the cloud service provider to their consumer, and it includes some elements that add additional value to consumers, such as Quality of Service (QoS) values and technical support details [5]–[7]. The cloud service providers are providing their services offers via their websites, whereas these website schema and layout vary from that provider to another. For example, the service offering template in Microsoft Azure marketplace website is unlike the one presented in the Amazon cloud web marketplace website [8], [9]. Therefore, one of the challenges that the cloud consumers face is to have an optimal cloud service discovery framework to guide them in finding a suitable and trustworthy cloud service from different cloud service advertisement on the web.

Cloud service discovery (CSD) is emerging as a new trend for service discovery across distributed and heterogeneous environments online. It is a process for locating a cloud service that best matches the end-user's requirements. Since the emergence of cloud technologies, cloud providers advertise their services online, and end-users make use of general search engines such as Bing and Google to discover cloud services [10]. The ability to explore cloud services advertisements across multiple websites becoming a challenge for cloud consumers, mainly when there is a large marketplace of those services. Many Web portals contain up to date cloud services advertisements such as getApp. These advertisements can be extracted and analysed using different web harvesting technique. Cloud consumers usually get confused by the massive number of possibly irrelevant search results due to keyword-based searches.

There is many literatures [10]–[14] in which the authors tried to address the issue. Nabeeh *et al.* [13] suggested adding a semantic annotation to cloud service profiles online to automate the discovery of cloud services. The objective of using semantic annotation is to allow search engines (such as Google) to semantically identify and retrieve service information based on a user's objectives [10]. A key issue with the semantic-based approach is that the semantic search could vary depending on the ontology domain and terminologies covered [15]. Alkalbani and Hussain [11] conducted a survey and found that almost all of the existing studies reuse an existing ontology. Such as a business ontology to semantically describe cloud service functions and improve query precision [12], [16]. Constructing an ontology which contains all the relevant domain concepts, such as service classification, service type, etc., is not an easy task, given the fact that cloud providers use different terminologies and vocabularies to describe their service offers, even though they have the same features [14]. Akinwunmi *et al.* [2] addressed the issue by proposing a decentralized agent system which acts as a consultant for cloud consumers to improve their experiences with cloud services. However, this system, like other approach makes use of a web search engine to find the services, and it is still in the conceptual phase without enough practical applications in the real environment.

Although existing literature has expended a great effort on enhancing search engines techniques with semantic annotations or by developing semantic-based systems for discovery cloud services, however, none of the existing literature discussed a centralised cloud services repository framework that has the capacity to extract, integrate and store cloud services advertisements from semi-structured data located in multiple and poorly organised Web portals into a centralized repository. The centralised repository plays a vital role in an efficient and effective service advertisement discovery in WWW that has been ignored. In this paper, we tried to address the issue by proposing the CCSR framework. The proposed framework presents how to harvest cloud services advertainments from the web portals using our developed harvesting tool - Harvesting-as-a-Service (HaaS). Using HaaS consumers crawl data from the WWW without doing any coding and retrieve relevant details in a fractional of time compared to the traditional methods. The extracted real-time data can be transferred in any format such as CSV, or pdf depending on the choice of a user. The data then analysed and mapped into a central repository that is used for cloud service discovery.

Significance of the Paper: This study is significant from the following two perspectives. Firstly, the harvesting tool 'HaaS 'allows users to extract real-time data from the web without the need to write any code. Secondly, the proposed centralised cloud service repository enables a consumer for efficient and effective cloud service discovery. The centralised repository act as a knowledge source for cloud services. Thirdly, we didn't find free real cloud dataset on cloud services. This study provides real cloud services dataset and actual cloud reviews that could be used by potential cloud consumers and for future research as well.

The rest of the paper is organized as follows: In Section 2 we discuss and critically analyse the related studies, Section 3 describes the proposed system architecture, Section 4 outlines the system workflow, Section 5 presents the system implementation, Section 6 conducts the system evaluation and presents the results and discussion, and the work concludes with a summary and description of future work in Section 7.

II. RELATED STUDIES

This section aims to provide a background to the state-of-the-art cloud service discovery approaches by studying how these approaches have been examined in the existing literature. The section analyses each of the contexts, features and methods for each of the approaches. We summaries some of the approaches and highlight gaps at the end of the section.

In the current literature, approaches in terms of service selection and composition, trust model [17] and reputation-based approaches [18] are the main approaches in the cloud computing context as well as in other domain such as Web Services and mobile services [19]–[23]. Also, machine learning technologies have been involved in the area of cloud computing to enhance the cloud service level of agreement [21], [24]–[28]. The cloud service discovery process depends on the technique and methods applied and used

to allocate the service information online. One of the most popular methods for service discovery task across all domains and industries is Google. Google is not, however, restricted to finding service information. Therefore, it may retrieve relevant, irrelevant information depending on the descriptive information provided [29]. In addition, searching for the service information using web search engines depends on two factors: (1) the keyword that best indicates the target service which is the most important aspect in internet marketing and the web search engine's algorithm; and (2) finding a service that is related to cloud services that best match user's requirements. The latter can be challenging if the cloud marketplace is vast. In addition, there are a considerable number of websites relating to non-existing services. Parhi *et al.* [30] proposed a semantic framework for cloud service description based on a multi-agent approach to support the location of cloud services [16]. The proposed framework assist consumer for the discovery of cloud services by referring to shared cloud ontology taxonomies to allocate an appropriate service. Kang and Sim [31] developed a cloud service discovery system (CSDS) that assists cloud users in finding cloud services over the Internet. The framework comprises a user interface and three agents. The model consults a cloud service ontology, which comprises a taxonomy of cloud service concepts. The user interface allows end-users to enter a query which specifies their preferences, including a service name and service requirements. To search for the service over the Internet, the query processing agent uses the existing web search engine - Google. Although the proposed system assists the consumer in finding cloud services, however, the system does not have a centralised repository for effective result. In another work, Sim [32] proposed a multi-agent cloud service discovery system Cloudle that focuses on matching end-users' requirements with an advertised cloud service. The proposed system comprises of four self-organising agents that are used to assist users in finding advertised cloud services and managing cloud resources. It considers variations in the types of cloud service resources over the Internet and consults cloud ontology taxonomies in the search process to match services and requirements. One of the shortcomings of this work is that the cloud search engine agent deployed to gather cloud services information uses a web search engine, which is a keyword-based search engine that also retrieves irrelevant information. To overcome the issue of key-word based searching, Rajendran and Swamynathan [33] proposed a model that combines the advantages of a cloud service ontology technique with a multi-agent-based protocol. The approach uses cloud service ontology to discover and retrieve information about cloud service. Although the system has a user-friendly interface that supports the end-user to find appropriate cloud services. However, the approach, like other existing work, make use of search engine, such as Google, to retrieve cloud services from the Internet and lacking the concept of a centralised repository. The idea of the centralised repository proposed by Chen *et al.* [34]. The framework comprised of two modules - web service description language (WSDL) and web service registry (UDDI) [35]. The approach semantically annotate cloud services using WSDL extension [36] and then store the semantic annotation of the cloud service in a web service registry - UDDI [37]. Although the proposed system provides a solution for dynamic cloud services selection; however, the approach is unable to cope with the growing marketplace and how to update in a real-time manner. Building the concept of a centralised repository, Alkalbani *et al.* [38] proposed a centralized open-source repository for cloud services. The approach is supported by the Nutch Hadoop Crawler [39], [40] which crawls a Web portal to extract information about cloud services and stores this information in a local repository. The authors only focused on SaaS service data and based on crawling information from one web portal only. To gather a variety of cloud services detail for different services, Gong and Sim [41] developed three versions of a cloud crawler. The proposed crawler gathers cloud service information from three well-known cloud providers - Amazon Web Services [42], Rackspace and GoGrid. The gathered data has two tuples, service specification and service price. Further analysis was applied to the service specification using a K-means clustering algorithm to cluster cloud services in a different category. One of the shortcomings of the proposed approach is that the crawler needs to be customized for every website. It, therefore, failed to crawl cloud services efficiently, since customizing the crawler is a time-consuming process. Also, there are some other studies on mobile service selection discussed the importance of service selection, including quality of service parameters in selecting the mobile services [43]–[45].

The above-discussed approaches tried to enhance cloud service discovery. However, there are many shortcomings in those approaches which are discussed as follows:

- The existing literature did not give an optimal solution of how to extract, integrate and store cloud services details from different semi-structured data spread across the internet.
- The discussed approaches did not provide any solution for customised harvesting tool while collecting cloud services information from heterogeneous web sources to build a comprehensive listing of cloud services.
- The existing literature did not discuss the harvester that extracts real-time data from the web without the need to write any code.
- The discussed approaches did not provide a means for integrating all cloud services information in a central repository for efficient and effective decision making.

To overcome the discussed gaps, we present our proposed CCSR framework, as discussed in Section 3.

III. CCSR FRAMEWORK

In this section, we propose the architecture of the Centralized Cloud Services Repository (CCSR) framework, which acts as a directory for cloud services advertisements and assists in efficient finding for cloud services commercial offerings. The CCSR framework composed of two modules: Harvesting

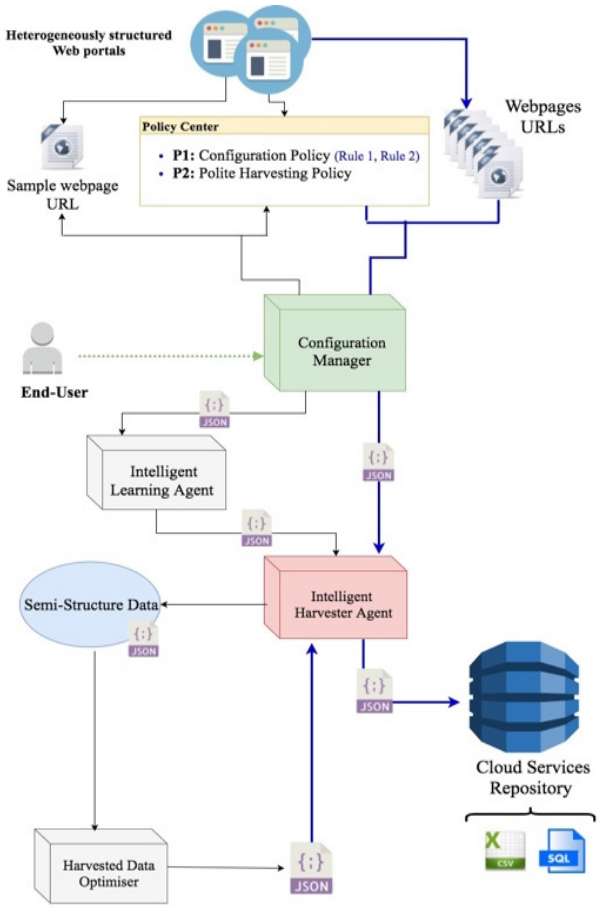

FIGURE 1. Centralized Cloud Services Repository (CCSR) framework.

as-a-Service (HaaS) and the Service Repository, as shown in Fig. 1. The detail of each module is discussed below:

A. HaaS MODULE

This section presents the design of our Harvesting as a Service (HaaS) module, which harvests cloud services information from targeted Web portals. The proposed HaaS harvest the real-time data/information from the targeted website in a few minutes and make it available in one file. Depending on the choice of a consumer, the file can be available in JSON, CSV, SQL or PDF file formats. Besides, the HaaS has a user-friendly interface that makes the process of harvesting easy for the end-users. By ''end-user'', we mean a cloud end-user such as a consumer, organization, or developer who has knowledge about cloud services advertisements and where to find them over the WWW. The HaaS comprised of six sub-modules - *Policy Centre, Configuration Manager, Learning Agent, Harvester Agent, Semi-structured Harvested Data, Harvesting Optimizer and Cloud Services Repository.*

The sub-modules are explained below:

1) POLICY CENTRE

The policy centre defines P1-configuration policy and P2- the polite harvesting policy for coordinating the harvesting process. Initial policies are set in P1 that outlines the boundary for carrying out the harvesting process. It provides details about the structure of the service advertisement in the webpage and the targeted service information to be harvested. The P1 work based on two rules. The first rule assists to choose sample page from the target website. The second rule then identifies the data that need to be harvested from that sample page. In our scenario, we have a list of service advertisement attributes (custom attributes), such as service name, review etc. and its organisation into object attributes such as review date, review name etc. The second policy P2 regulates the maximum time for each harvesting session. To avoid overloading, the harvesting process is divided into multiple sessions, depending on the total number of related webpages. After each session, the process is paused for a certain period before continuing a new session. In this way, the harvested targets website is not overloaded. The user indicates how many links can be harvested per session and indicate how many seconds to wait before moving on to the next session.

2) CONFIGURATION MANAGER

The Configuration Manager personalizes the harvesting process and provides essential guidelines for collecting service information form the targeted website. The harvesting process is comprised of two phases: phase one is the setup phase and phase two is the harvesting phase. Phase one is conducted in four steps, using the Configuration Manager user interface. During this setup phase, the Configuration Manager utilizes three levels of the configuration structure: the web page level, the custom object level, and the custom attribute level. The web page level provides a sample Uniform Resources Locator (URL) of a web page, while the custom object and the custom attribute levels define the data targeted for collection. The Configuration Manager consults the Policy Centre to ensure defined rules/policies are followed during the procedure.

3) LEARNING STRUCTURE AGENT

The task of the Learning Agent is to learn the web page layout/structure of the target Web portal. We propose an algorithm for learning the HTML structure of a web page, as shown in Algorithm 1. It is a learning algorithm that required the user to determine specific control parameters such as targeted web page URL and the required information from in the targeted web page. Sample metadata for a particular page in the targeted website collected and stored in JSON file format. In the sample date, we need to indicate the following: indicates the URL of the sample page, indicates list of objects of the sample page (S), only get the first 25 characters of a sample string, find all HTML tags which have a specific text, Get name and some attributes of a HTML tag (O), Get name, some attributes of a HTML tag and its position compared with other tags of the same type under a HTML parent tag (a). The output of this algorithm is the metadata structure in JSON format with the core is the configuration data with extended information to navigate the position of attributes

Algorithm 1 Learning structure Algorithm

- Input: Configuration data in JSON format following the structure:
- $S = \{multiple, objects = (o_1, o_2 \ldots o_n)\}$ is the structure of the configuration, containing a sequence of configured objects, $multiple \in \{yes, no\}$ (get one or multiple HTML data for objects of the same structure)
- o_i $(i \in N) = \{attributes = (a_1, a_2 \ldots a_m)\}$ is a detail content of each object including a sequence of configured attributes.
- a_j $(j \in \{1, 2, 3 \ldots m\}) = \{sample, multiple, fullText\}$: *sample* (sample text for the attribute), $multiple \in \{yes, no\}$ (get one or multiple HTML data of similar HTML sibling tags) and $fullText \in \{yes, no\}$ (sample text is in full content or partial content)

Output: Metadata structure in JSON format with the core is the configuration data with extended information to navigate the position of attributes during the harvesting process. The output for objects and attributes is as follow:

- o_i $(j \in N) = \{attributes = (a_1, a_2 \ldots a_m)$, *parentTag* $= \{position, tagName, className, idName\}\}$. *Position*: the position of o_i *tagName* in relation to the whole HTML document, *tagName*: HTML tag name, *className*: HTML class name, *idName*: HTML ID name.

$a_j = \{sample, multiple, fullText,$ *filterTag*$=\{position, tagName, className, idName\}\}$. *Position*: the position of the *tagName* in relation to the *parentTag* of the object containing a_j, *tagName*: HTML tag name, *className*: HTML class name, *idName*: HTML ID name.

Procedure: Begin Algorithm
For i = 1 to n
  Fetch the sample of the first attribute a_1 in o_i
  Compute HTML parent tag of a_1 and stores *parentTag* in o_i using BeautifulSoup
  For j = 1 to m
    If *parentTag* does not contain a_j then
      Compute HTML parent tag and then store *parentTag* in o_i using BeautifulSoup
      Repeat step 4
    End if
  End for
  For j = 1 to m
    Compute HTML tag of a_j based on computed *parentTag* in o_i and stores *filterTag* in a_j using BeautifulSoup
  End for
End for

End Algorithm

during the harvesting process. The output for objects and attributes is as shown in the Algorithm 1 Output section.

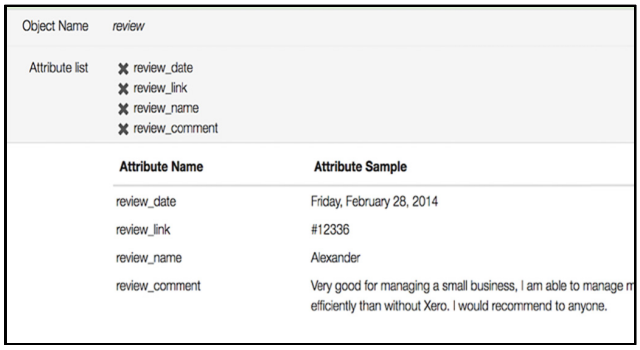

FIGURE 2. Harvested sample data.

The output also displayed to the end-user for data validation and modification, as showed in Fig 2. The end-user verification is to ensure that the sample data is correct and complete. Steps 1, 2 and 3 are a recursive process until the end-user-defined harvesting boundary has been reached.

4) WEB PAGE HARVESTER ALGORITHM

The task of this component is to extract meaningful information about cloud services from the Web, as specified by the end-user in the data configuration step. To handle the heterogeneity structured of service information when dealing with a large number of web pages, we define policies within the Policy Centre. The Harvester Agent then carries out the harvesting process based on the defined policies. The algorithm pseudocode for Harvester Agent shows in Algorithm 2. The input in this algorithm is the JSON file output from Algorithm 1, that has sample metadata. It shows a list sample metadata of all sample pages and indicates the pattern of the URL of the sample page. The output of this algorithm is the JSON file that has a list of the harvested information, as presented in Figure 2.

5) SEMI-STRUCTURED HARVESTED DATA

This component responsible for receiving the structured harvested information coming from the Web Page Harvester. This information is structured as a JSON object, using the restrictions specified by the end-user during the configuration phase. At this stage, the file includes harvested information with some redundant service attributes.

6) HARVESTING OPTIMIZER

The objective of this component is to remove redundant service attributes from the harvested information file. These attributes are useful for learning the correct HTML structure of the sample targeted web page. The redundant attributes removed after the learning process. To remove the redundant attributes the user assesses the sample harvested information file, then the user can remove any redundant attribute via pressing a delete button in the user interface. It contains all the cloud service information, including cloud services, offers details such as service Uniform Resource Locator (URL), service name, service type, service category, and details of

consumers' reviews such as reviewer name and comments. This information is made available in different formats such as CSV, PDF or SQL.

B. SERVICES REPOSITORY MODULE

This component is responsible for storing and mapping the harvested information, which has meaningful information about the cloud services advertisement. To achieve this, we use ontology to represent the knowledge of the stored information. The reason for choosing the ontology is because it provides a shared and common understanding of a cloud services advertisement that communicates between people across different web platforms [46]. For example, Amazon (www.amazon.com) and eBay (www.ebay.com) have used the ontologies in products classification for sales and their features [47]. Therefore, we construct the cloud services advertisement ontology for the purpose of the sale by referring to the NIST classification for cloud services which includes three main categories: SaaS, PaaS, IaaS [48], as shown in Fig 3. This ontology offers a first conceptual of the knowledge of cloud services advertisement. Then, we map and store each cloud services advertisements into a concept in the ontology. We extracted meaningful information into the SQL file that represents the main attributes of each cloud service advertisement. We consider that cloud service advertisement 'A' is represented by service metadata M, which describes the general knowledge of the cloud service commercial offer such as service ID, service name and service details. In this work, we utilise the service metadata descriptive information from the relational database, which has the harvested information organised by attributes, such as service name, service description and service category. To identify the service concepts that are relevant to a particular service category concept, [*service id, service name, service category, service description, provider link, free trial (yes, no), mobile app (yes, no), rating, starting price, year founded. Service ID:* is the URI of the service, which is the reference to the semantically linked concepts. *Service Name:* is the name of the service, *Service Category:* is the category to which the service belongs, *Service description:* is the detailed text description of the service features and facilities. *Provider link:* is the URL link of the service provider, *Starting Price:* is the starting price of the service per month, *Rating:* is the score that a consumer gives to a service after purchasing and using it, *Free Trial:* indicates if the service is available for free a trial or not, *Mobile App:* indicates if the service is a mobile application or not.

IV. EVALUATION AND VALIDATION OF CCSR FRAMEWORK

To validate the proposed CCSR framework, we demonstrate a case study shows establishing of cloud services central repository using web harvesting tool (HaaS). Our intention in this case study is to show how the CCSR framework assists in establishing a central repository for cloud services advertainments, which assist in discovering cloud

Algorithm 2 Harvest URLs Algorithm

Input: The algorithm Harvest URLs has three types of input:

1. Metadata structure from the output of Algorithm 1: Structure algorithm
2. A sequence of URLs that have the same HTML structure with the configured Web page. URL = (url_1, $url_2 \ldots url_p$)
3. Polite harvesting parameters {*recordsPerSession*, *waitingTimeInterval*}. *recordsPerSession*: the number of URLs to be harvested over one session, *waitingTimeInterval*: the time (seconds) the harvest process pause between each session.

Output: Dataset includes data of configured attributes for all inputted URLs. The dataset is stored in MongoDB and exported to CSV format.

Procedure: Compute the number of sessions NS based on the number of URLs and parameter *recordsPerSession.* NS = Number of URLs/*recordsPerSession*
Set session to 1
While session <= NS then
  For k = 1 to p
    For i = 1 to n
      Compute all possible *parentTag* of o_i inside url_k using BeautifulSoup and store these tags into ts
      If *multiple* of o_i is "yes" then
        Repeat step 12 to step 18 for all *parentTag* inside ts
      Else then
        Perform step 12 to step 18 for the *parentTag* that has the position aligned with o_i position
      End if
      For j = 1 to m
        If *multiple* of a_j in o_i is "yes" then
          Parse the content of a_j for all similar HTML siblings based on *parentTag* and a_j *filterTag* (*tagName*, *className*, *idName*) using BeautifulSoup and store the data in MongoDB for url_k
        Else then
          Parse the content of a_j for the HTML tag based on *parentTag* and a_j *filterTag* (*position*, *tagName*, *className*, *idName*) using BeautifulSoup and store the data in MongoDB for url_k
        End if
      End for
    End for
  End for
  Pause the process based on *waitingTimeInterval*
  Increment session to 1
End while

End Algorithm

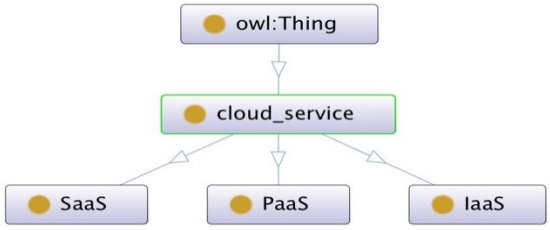

FIGURE 3. Cloud services advertisement ontology.

TABLE 1. Comparison of Crawly, Parsers, HaaS and Scrapy.

Comparison Aspects	Scrapy	Crawly	HaaS	Parsers
Type of services	A	S	A	E
Extracted data format	JSON, CSV and XML	JSON and CSV	JSON and CSV	JSON, CSV and XML
User Interface	X	X	√	X
Number of sites for scarping	Unlimited	Limited	Unlimited	Limited based on subscription
Customize the crawler using coding	√	√	X	X
Polite Harvesting feature	√	X	√	X

√: denote that the tool includes the aspect
X: denote that the tool does not include the aspect
A: denote that the tool is an application
S: denote that the tool is a site
E: denote that the tool is an extension

services information. For this study, we consider harvesting cloud services advertainments details from two publicly available web portals, namely getapp.com and serchen.com. To demonstrate the feasibility of our proposed 'HaaS framework', we compare it with the other three widely used open-source harvesters – Parsers, Crawly and Scrapy as presented in Table 1. We compare them based on six criteria that we believe are an essential factor for user efficiency and satisfaction. From the comparative analysis, we see that there are many similarities between HaaS and the other harvesters. However, our approach does not need the end-users to code for harvesting, and it provides a user-friendly interface that makes it usable for the end-users to easily employ the harvesting process.

To harvest serchen.com, the HaaS system takes the end-user through some steps discussed as follows:

A. CONFIGURATION

The purpose of this step is to help users to configure what they want to harvest on their target Web portal. First, the user needs to investigate serchen.com thoroughly to understand the serchen.com sitemap and the layout of serchen.com web pages with repetitive HTML structure that they wish to focus. To set up the configuration, users need to provide the

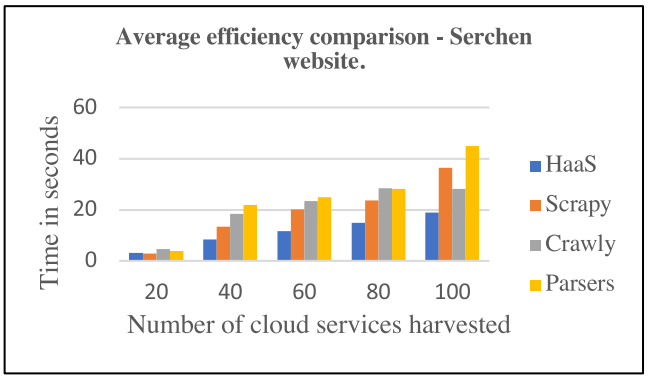

FIGURE 4. Average efficiency comparison - Serchen website.

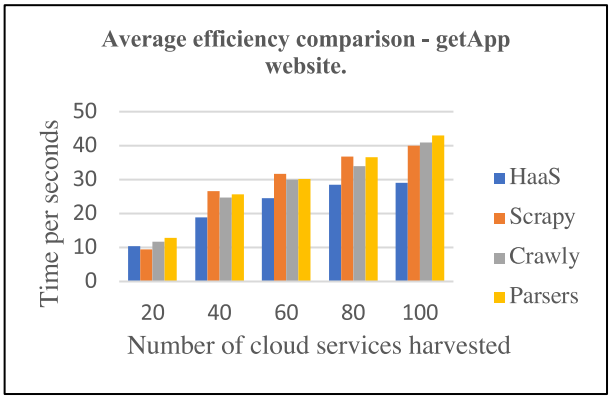

FIGURE 5. Average efficiency comparison - getApp website.

following details on the configuration form. PAGE has Website URL: Original URL of the harvested website and Page URL: Sample page URL to obtain sample data. A page can have one or many objects. OBJECT has Object Name: Custom object name. An object can have one or many attributes, and Object Multiple: This option indicates whether to harvest the same object multiple times. ATTRIBUTE has Attribute Name: Custom attribute name, Attribute Sample: Attribute sample copied from the sample page. If the text is too long, partial text used, Multiple: Indicate whether to harvest the same attribute multiple times (e.g. harvest multiple li in an ul tag) and Full text: Indicate whether the attribute sample is in full text or partial text (check HTML structure. The Add Object button (+ object) inside the page panel is used to add a new empty object to the Configuration page. The Add Attribute button (+ attribute) inside the Object panel for the collection of attributes is used to add a new empty attribute to the Object on the Configuration page. Users can remove Objects/Attributes using the delete buttons. Clicking the Next button takes users to Step 2: View Sample. The screenshot in Figure 7 shows the configuration setup screen for serchen.com based on sample pre-defined data from the system.

B. VIEW SAMPLE

the purpose of this step is to learn the HTML structure of all configured objects and their attributes in the configured page. It also assists users to validate the results to ensure

that the required data is sampled, per the user's request at step 1: Configuration. If the data is not correct, users can move back to Step 1 using the Previous button to adjust their configuration of the error object/attribute. If this is the case, users click the button to trigger the system to re-learn the structure and re-get the sample data. Removing redundant attributes to customize the harvested result is carried out in Step 4: Start Harvesting. Users click the Next button to move to Step 3: STEP 3: RELATED LINKS. THIS STEP ASSISTS users to copy the URLs of all web pages that have the same HTML structure as the configured sample web page from Step 1: Configuration. The cloud services offerings located in a unique web page URL, but all pages have the same structure. The polite harvesting feature implemented in the HaaS system with constraints that include records per session and waiting for a time interval (refer to Section 3). Users click the Next button to move to the next step, Start Harvesting.

Start Harvesting the users only need to press the Start Harvesting button to start the harvesting process. The system stores harvested information in MongoDB with JSON syntax. When the harvesting process finished, the system generates data in CSV file format and makes it available to the end-user. The web browser pops up another Tab that enables users to download the data in CSV file format (save the file name as <filename>.csv). If automatic popups blocked on the user's browser, the user needs to allow popups for the web tool and re-start harvesting to download the file. The structure of the exported file is as follows: All Attributes which belong to an Objects of multiple values and assigned to No, they are combined and considered as columns in the top table (Main Table of output Datasets). To harvest the getapp.com Web portal, we followed the same steps. Endusers follow the HaaS user interface instructions to harvest the data from the target Web portals.

C. CONSTRUCTING ONTOLOGY AND REPOSITORY

In this step, the Web ontology language (OWL) used to represent the conceptual model and the knowledge of the collected cloud services advertainments. We extracted meaningful information into the SQL file that represents the main attributes of each cloud service advertainment. We consider that cloud service advertisement 'A' is represented by service metadata M, which describes the general knowledge of the cloud service commercial offer such as service ID, service name and service details. In this work, we utilise the service metadata descriptive information from the relational database, which has the harvested information organised by attributes, such as service name, service description and service category. To identify the service concepts that are relevant to a certain service category concept, [service id, service name, service category, service description, provider link, free trial (yes, no), mobile app (yes, no), rating, starting price, year founded].

Service ID: is the URI of the service, which is the reference to the semantically linked concepts. Service Name: is the name of the service, Service Category: is the category to which the service belongs, Service description: is the detailed text description of the service features and facilities. Provider link: is the URL link of the service provider, Starting Price: is the starting price of the service per month, Rating: is the score that a consumer gives to a service after purchasing and using it, Free Trial: indicates if the service is available for free a trial or not, Mobile App: indicates if the service is a mobile application or not. The ontology has been implemented and tested using Protégé Software [49].

TABLE 2. Harvesting time.

Targeted Web portal	No. of harvested cloud services advertainments	No. of harvested cloud reviews
serchen.com	12511	11078
getapp.com	5295	6259
Total	17806 services	17337 reviews

TABLE 3. Efficiency comparison (in seconds) - Serchen website.

Harvesting Tool	Number of services harvested				
	20	40	60	80	100
	Harvested Cloud services (1st round)				
HaaS	*3.09*	*8.5*	10.9	15.5	19.13
Scrapy	*2.31*	*17.4*	21.4	17.6	43.12
Crawly	*4.65*	*19.3*	*25.8*	*22.3*	*41.5*
Parsers	*4.5*	*21.04*	*25.2*	*22.1*	*41.43*
	Harvested Cloud services (2nd round)				
HaaS	3.06	8.3	11.15	14.5	18.05
Scrapy	3.03	10.5	21.9	29.8	50.02
Crawly	4.16	20.7	24.7	33.03	49.3
Parsers	3.44	24.05	27.46	30.23	52.17
	Harvested Cloud services (3rd round)				
HaaS	3.13	8.45	11.63	14.98	18.81
Scrapy	2.9	13.45	20.17	23.66	36.35
Crawly	4.8	23.89	27.92	30.07	43.64
Parsers	3.67	20.88	22.07	31.9	41.2
The average efficiency of three rounds					
HaaS	3.13	8.45	11.63	14.98	18.8
Scrapy	2.9	13.45	20.17	23.66	36.35
Crawly	4.53	18.29	23.47	28.46	39.15
Parsers	3.86	21.9	24.92	28.07	44.93

D. RESULT

The HaaS was able to generate of 17657 cloud services items and 17337 consumers reviews experience with cloud services as presented in Table 2.

In addition we examine the efficiency of the proposed HaaS system compared to the Parsers, Scrapy and Crawly heterogeneous structured websites. We first harvested the *serchen.com* Web portal using all of them without applying the polite harvesting feature and compared both tools about crawl time. Tables 3 shows this comparison of harvesting time across three rounds of harvesting. To validate the harvesting results, we measured the percentage of change in the

TABLE 4. Efficiency comparison (in seconds) - getApp website.

Harvesting Tool	Number of services harvested				
	20	40	60	80	100
	Harvested Cloud services (1st round)				
HaaS	8.39	18.8	16.95	26.72	29.04
Scrapy	7.48	20.72	29.71	30.58	38.88
Crawly	*8.03*	*17.53*	*25.81*	*28.45*	*41.77*
Parser	*8.52*	*17.02*	*25.77*	*28.8*	*41.03*
	Harvested Cloud services (2nd round)				
HaaS	8.2	18.45	26.9	28.06	29.38
Scrapy	7.64	28.68	32.93	39.5	38.2
Crawly	8.65	25.82	33.32	33.9	40.04
Parsers	10.83	27.14	33.54	38.09	44.25
	Harvested Cloud services (3rd round)				
HaaS	14.7	19.23	30.03	30.98	28.65
Scrapy	13.3	30.47	32.38	40.2	42.7
Crawly	18.6	30.85	30.99	39.64	40.9
Parsers	19.04	32.75	31.05	42.9	43.71
The average efficiency of three rounds					
HaaS	10.43	18.83	24.62	28.58	29.02
Scrapy	9.47	26.63	31.67	36.76	39.92
Crawly	11.76	24.73	30.04	33.99	40.9
Parsers	12.92	25.6	30.12	36.6	42.99

TABLE 5. Polite harvesting details.

Harvesting Tool	Record per session	Waiting time interval (in secs)
Scrapy	10 pages	3
Crawly	10 pages	4
HaaS	10 records	3
Parsers	10 pages	6

harvesting time as a function of the number of harvested services (20, 40, 60, 80 and 100). The harvesting results of serchen.com show that the proposed HaaS tool performs better than Parsers, Scrapy and Crawly.

The harvesting time usually depends on network bandwidth, CPU capacity at the time of running, server response time at the time of running, and the polite harvesting configuration for all HaaS, Scrapy, Parsers and Crawly. Tables 4 presents the comparison of harvesting time for HaaS, Crawly and Scrapy across three rounds. We next harvested getapp.com using HaaS, Crawly, Parsers and Scrapy. We applied the polite harvesting feature for both tools, as presented in Table 5.

Also, we tested the quality of the harvested data using our approach compared with the tradition tool, such as scrapy. We have compared the results of harvesting 100 cloud services from serchen.com using Scrapy and HaaS. The Fig 4 and Fig 5 present the harvested data in two columns service URL, service name.

The results indicate that significant values were missing from the service description column in the case of the Scrapy results. By ''missing value'', we mean that the corresponding value was not present in the harvested data as presented in Figure 6 and Figure 7. There are 14 missing service name values out of 100 harvested services, with a successful

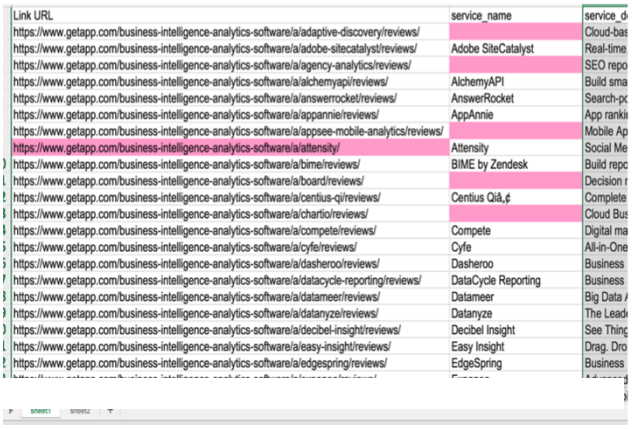

FIGURE 6. Screenshot of data harvested from serchen.com by Scrappy.

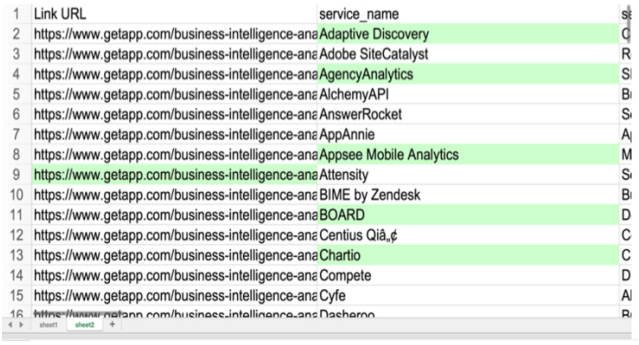

FIGURE 7. Screenshot of data harvested from serchen.com by HaaS.

TABLE 6. Experiment results mapping cloud services ads entity to the CSA ontology.

Concept type	Number of concepts correctly instantiated	Number of concepts not correctly instantiated
Instances	17,793	13
Data Type	124,551	91
Total	142344	104

harvesting quality rate of 86%. Interestingly, there are no missing values in the HaaS file (Fig 5), indicating the successful harvesting quality rate of 98%.

We used precision, recall, and F-score methods to tagged extractions about the instantiated concepts in the knowledge base as presented in the below equations.

$$Precision = \frac{number\ of\ instances\ correctly\ acquired}{number\ of\ instances\ acquired}$$

$$Recall = \frac{number\ of\ instances\ correctly\ acquired}{number\ of\ instances\ existing\ in\ the\ conceptual\ tree}$$

$$F-measure = \frac{2 * recall * precision}{recall + precison}$$

The harvested 17806 service entities are collected from three different web portals in the repository using the equations mentioned above. Table 5 presents the results of the set of individuals and concepts extracted from the repository to map and populate the cloud service advertainments (CSA) ontology. A set of 17806 individual mappings according

TABLE 7. Performance result.

Concept Type	Precision	Recall	F-measure
Instances	99.87	99.06	99.46
Data type properties	99.32	98.91	99.11
Total	99.59	98.98	99.28

TABLE 8. Performance of Service advertisement retrieval using ontology.

	Recall	Precision	F-measure
Service advertainment retrieval	99.59%	98.98%	99.28

to the conceptual tree CSA produced. Of these concepts, 17793 correctly instantiated by the rules of the classification (SaaS, PaaS and IaaS) and 13 not instantiated. We thus obtain a recall of 99.59 and precision of 98.98. These accuracy measures are shown in Table 8.

V. CONCLUSION

In this study, we have presented a service-based harvester called *Harvesting as a Service* (HaaS) for crawling cloud services information from various structured Web portals. The critical contribution of the proposed system is the HaaS with a friendly user interface, which allows end-users to harvest websites without the need for developers or coding, unlike other traditional harvesting tools. Experiments were carried out, and the results show that compared to the traditional tool, our proposed approach demonstrates a significant improvement in the harvesting time quality when the number of harvested pages increased. Also, we used an ontology for storing and representing the harvested data. Future work will consider continuing harvesting of the various web portals to enrich the cloud service advertainments ontology.

REFERENCES

[1] W. Hussain, F. K. Hussain, O. Hussain, and E. Chang, "Profile-based viable service level agreement (SLA) violation prediction model in the cloud," in *Proc. 10th Int. Conf. P2P, Parallel, Grid, Cloud Internet Comput. (3PGCIC)*, Nov. 2015, pp. 268–272.

[2] A. O. Akinwunmi, E. A. Olajubu, and G. A. Aderounmu, "A multi-agent system approach for trustworthy cloud service discovery," *Cogent Eng.*, vol. 3, no. 1, 2016, Art. no. 1256084.

[3] L. Sun, H. Dong, F. K. Hussain, O. K. Hussain, and E. Chang, "Cloud service selection: State-of-the-art and future research directions," *J. Netw. Comput. Appl.*, vol. 45, pp. 134–150, Oct. 2014.

[4] W. Hussain, F. K. Hussain, O. K. Hussain, E. Damiani, and E. Chang, "Formulating and managing viable SLAs in cloud computing from a small to medium service provider's viewpoint: A state-of-the-art review," *Inf. Syst.*, vol. 71, pp. 240–259, Nov. 2017.

[5] W. Hussain, F. K. Hussain, M. Saberi, O. K. Hussain, and E. Chang, "Comparing time series with machine learning-based prediction approaches for violation management in cloud SLAs," *Future Gener. Comput. Syst.*, vol. 89, pp. 464–477, Dec. 2018.

[6] Y. Yin, *QoS Prediction for Service Recommendation With Deep Feature Learning in Edge Computing Environment*. New York, NY, USA: Springer, 2019, pp. 1–11.

[7] Y. Yin, W. Xu, Y. Xu, H. Li, and L. Yu, "Collaborative QoS prediction for mobile service with data filtering and SlopeOne model," *Mobile Inf. Syst.*, vol. 2017, Jun. 2017, Art. no. 7356213.

[8] *Amazon Web Services—Cloud Computing Services*. Accessed: Aug. 18, 2019. [Online]. Available: https://aws.amazon.com/

[9] *Microsoft Azure Cloud Computing Platform & Services*. Accessed: Aug. 18, 2019. [Online]. Available: https://azure.microsoft.com/en-us/

[10] A. Goscinski and M. Brock, "Toward dynamic and attribute based publication, discovery and selection for cloud computing," *Future Gener. Comput. Syst.*, vol. 26, no. 7, pp. 947–970, 2010.

[11] A. M. Alkalbani and F. K. Hussain, "A comparative study and future research directions in cloud service discovery," in *Proc. IEEE 11th Conf. Ind. Electron. Appl. (ICIEA)*, Jun. 2016, pp. 1049–1056.

[12] F. Gong, Y. Ma, W. Gong, X. Li, C. Li, and X. Yuan, "Neo4j graph database realizes efficient storage performance of oilfield ontology," *PLoS ONE*, vol. 13, no. 11, 2018, Art. no. e0207595.

[13] N. A. Nabeeh, A. El-Ghareeb, and A. M. Riad, "Review of cloud services discovery," *Adv. Inf. Sci. Service Sci.*, vol. 7, no. 2, pp. 28–39, Apr. 2015.

[14] L. Youseff, M. Butrico, and D. D. Silva, "Toward a unified ontology of cloud computing," in *Proc. Grid Comput. Environ. Workshop*, Nov. 2008, pp. 1–10.

[15] D. Bonino, F. Corno, L. Farinetti, and A. Bosca, "Ontology driven semantic search," *WSEAS Trans. Inf. Sci. Appl.*, vol. 1, no. 6, pp. 1597–1605, 2004.

[16] Y. Yin, L. Chen, Y. Xu, and J. Wan, "Location-aware service recommendation with enhanced probabilistic matrix factorization," *IEEE Access*, vol. 6, pp. 62815–62825, 2018.

[17] W. Hussain, F. K. Hussain, and O. K. Hussain, "Maintaining trust in cloud computing through SLA monitoring," in *Neural Information Processing*. Cham, Switzerland: Springer, 2014, pp. 690–697.

[18] A. Alghamdi, W. Hussain, A. Alharthi, and A. B. Almusheqah, "The need of an optimal QoS repository and assessment framework in forming a trusted relationship in cloud: A systematic review," in *Proc. IEEE 14th Int. Conf. e-Bus. Eng. (ICEBE)*, Nov. 2017, pp. 301–306.

[19] H. Gao, K. Zhang, J. Yang, F. Wu, and H. Liu, "Applying improved particle swarm optimization for dynamic service composition focusing on quality of service evaluations under hybrid networks," *Int. J. Distrib. Sensor Netw.*, vol. 14, no. 2, Feb. 2018, Art. no. 1550147718761583.

[20] L. Qi, W. Dou, W. Wang, G. Li, H. Yu, and S. Wan, "Dynamic mobile crowdsourcing selection for electricity load forecasting," *IEEE Access*, vol. 6, pp. 46926–46937, 2018.

[21] S. Pang, H. Chen, H. Liu, J. Yao, and M. Wang, *A Deadlock Resolution Strategy Based on Spiking Neural P Systems*. Berlin, Germany: Springer, 2019, pp. 1–12.

[22] L. Qi, J. Yu, and Z. Zhou, "An invocation cost optimization method for Web services in cloud environment," *Sci. Program.*, vol. 2017, May 2017, Art. no. 4358536.

[23] H. Gao, Y. Duan, H. Miao, and Y. Yin, "An approach to data consistency checking for the dynamic replacement of service process," *IEEE Access*, vol. 5, pp. 11700–11711, 2017.

[24] W. Hussain, F. K. Hussain, and O. K. Hussain, "SLA management framework to avoid violation in cloud," in *Proc. Int. Conf. Neural Inf. Process.* Cham, Switzerland: Springer, 2016, pp. 309–316.

[25] W. Hussain and O. Sohaib, "Analysing cloud QoS prediction approaches and its control parameters: Considering overall accuracy and freshness of a dataset," *IEEE Access*, vol. 7, pp. 82649–82671, 2019.

[26] W. Hussain, O. Sohaib, M. Naderpour, and H. Gao, "Cloud marginal resource allocation: A decision support model," in *Mobile Networks and Applications*. New York, NY, USA: Springer, 2019.

[27] W. Hussain, F. K. Hussain, and O. K. Hussain, "Risk management framework to avoid SLA violation in cloud from a provider's perspective," in *Proc. Int. Conf. P2P, Parallel, Grid, Cloud Internet Comput.* Cham, Switzerland: Springer, 2016, pp. 233–241.

[28] W. Hussain, F. K. Hussain, and O. K. Hussain, "Towards soft computing approaches for formulating viable service level agreements in cloud," in *Neural Information Processing*. Cham, Switzerland: Springer, 2015, pp. 639–646.

[29] C. Wu and E. Chang, "Searching services 'on the Web': A public Web services discovery approach," in *Proc. 3rd Int. IEEE Conf. Signal-Image Technol. Internet-Based Syst.*, Dec. 2007, pp. 321–328.

[30] M. Parhi, B. K. Pattanayak, and M. R. Patra, "A multi-agent-based framework for cloud service description and discovery using ontology," in *Intelligent Computing, Communication and Devices*. New Delhi, India: Springer, 2015, pp. 337–348.

[31] J. Kang and K. M. Sim, "Cloudle: An ontology-enhanced cloud service search engine," in *Proc. Int. Conf. Web Inf. Syst. Eng.* Berlin, Germany: Springer, 2010, pp. 416–427.

[32] K. M. Sim, "Agent-based cloud computing," *IEEE Trans. Serv. Comput.*, vol. 5, no. 4, pp. 564–577, 4th Quart., 2012.

[33] V. Rajendran and S. Swamynathan, "A novel approach for semantic service discovery in cloud using broker agents," in *Proc. Int. Conf. Adv. Comput., Commun. Inf. Sci.*, Jun. 2014, pp. 242–250.
[34] F. Chen, X. Bai, and B. Liu, "Efficient service discovery for cloud computing environments," in *Advanced Research on Computer Science and Information Engineering*. Berlin, Germany: Springer, 2011, pp. 443–448.
[35] D. Paulraj, S. Swamynathan, and M. Madhaiyan, "Process model-based atomic service discovery and composition of composite semantic Web services using Web ontology language for services (OWL-S)," *Enterprise Inf. Syst.*, vol. 6, no. 4, pp. 445–471, 2012.
[36] A. Ankolekar, M. Burstein, J. R. Hobbs, O. Lassila, D. Martin, D. McDermott, S. A. McIlraith, S. Narayanan, M. Paolucci, T. Payne, and K. Sycara, "DAML-S: Web service description for the semantic Web," in *The Semantic Web—ISWC*. Berlin, Germany: Springer, 2002, pp. 348–363.
[37] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana, "Unraveling the Web services Web: An introduction to SOAP, WSDL, and UDDI," *IEEE Internet Comput.*, vol. 6, no. 2, pp. 86–93, Mar./Apr. 2002.
[38] A. Alkalbani, A. Shenoy, F. K. Hussain, O. K. Hussain, and Y. Xiang, "Design and implementation of the Hadoop-based crawler for SaaS service discovery," in *Proc. IEEE 29th Int. Conf. Adv. Inf. Netw. Appl.*, Mar. 2015, pp. 785–790.
[39] Z. Laliwala and A. Shaikh, *Web Crawling and Data Mining With Apache Nutch*. Birmingham, U.K.: Packt, 2013.
[40] M. Olson, "HADOOP: Scalable, flexible data storage and analysis," *IQT Quart*, vol. 1, no. 3, pp. 14–18, Jan. 2010.
[41] S. Gong and K. M. Sim, "CB-Cloudle and cloud crawlers," in *Proc. IEEE 5th Int. Conf. Softw. Eng. Service Sci.*, Jun. 2014, pp. 9–12.
[42] AWS. *AWS Marketplace*. Accessed: Oct. 13, 2017. [Online]. Available: https://aws.amazon.com/marketplace
[43] Y. Yin, Y. Xu, W. Xu, M. Gao, L. Yu, and Y. Pei, "Collaborative service selection via ensemble learning in mixed mobile network environments," *Entropy*, vol. 19, no. 7, p. 358, 2017.
[44] H. Gao, D. Chu, Y. Duan, and Y. Yin, "Probabilistic model checking-based service selection method for business process modeling," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 27, no. 6, pp. 897–923, Feb. 2017.
[45] L. Qi, W. Dou, and J. Chen, "Weighted principal component analysis-based service selection method for multimedia services in cloud," *Computing*, vol. 98, nos. 1–2, pp. 195–214, 2016.
[46] G. Flouris, D. Manakanatas, H. Kondylakis, D. Plexousakis, and G. Antoniou, "Ontology change: Classification and survey," *Knowl. Eng. Review.*, vol. 23, no. 2, pp. 117–152, Jun. 2008.
[47] *The Product Types Ontology: Class Definition for, 'Amazon (Company)'*. Accessed: Aug. 19, 2019. [Online]. Available: http://www.productontology.org/doc/Amazon_%28company%29
[48] P. Mell and T. Grance, "The NIST definition of cloud computing," U.S. Dept. Commerce, Washington, DC, USA, Tech. Rep. NIST Special Publication 800-145, 2011.
[49] *A Free, Open-Source Ontology Editor and Framework for Building Intelligent Systems*. Accessed: Aug. 19, 2019. [Online]. Available: https://protege.stanford.edu/

ASMA MUSABAH ALKALBANI received the B.S. degree in computer engineering from the Caledonian College of Engineering, Muscat, Oman, in 2004, the M.S. degree in information technology from La Trobe University, Melbourne, Australia, in 2010, and the Ph.D. degree in software engineering from the University of Technology Sydney (UTS), Sydney, NSW, Australia. From 2010 to 2014, she was a Lecturer with the College of Applied Sciences, Oman, and from 2014 to 2018, and also a Casual Academician with the Computer Science School, UTS, where she supervised 12 Master Graduation projects (data analytics and service discovery). Since 2018, she has been an Assistant Professor with the Information Technology Department, College of Applied Sciences, Oman. Her research interests include service discovery, web data mining, and business data analytics to better understanding business need especially the data analytics of biomedical data, text mining, and social network analysis.

WALAYAT HUSSAIN received the Ph.D. degree from the University of Technology Sydney, Australia. He was a Lecturer and an Assistant Professor with the BUITEMS for many years. He is currently a Lecturer with the Faculty of Engineering and Information Technology, University of Technology Sydney. He has published in various top-ranked reputable ERA-A*, SJR-Q1, JCR-Q1 journals and conferences such as: *The Computer Journal* (Oxford University), the *Information Systems*, the *Future Generation Computer Systems*, IEEE ACCESS, the *Computers & Industrial Engineering*, the *Mobile Networks and Applications*, the *Journal of Ambient Intelligence and Humanized Computing*, the *Global Journal of Flexible Systems Management*, FUZZ-IEEE, and ICONIP. His current research interests include business intelligence, cloud computing, and usability engineering by focusing on providing an informed decision to different stakeholders. He was a recipient of National and International Research Awards and Recognitions. He was also the recipient of the 2016 FEIT HDR Publication Award by the University of Technology Sydney, Australia.

JUNG YOON KIM was born in Seoul, South Korea, in 1979. He received the B.S. and M.S. degrees from Hoseo University, in 2002 and 2006, respectively, and the Ph.D. degree from the Graduate School of Advanced Imaging Science, Multimedia & Film, Chung-Ang University, in 2013, all in game engineering. From 2004 to 2005, he was a Game Designer with Game Industry for online casual game. Then, he started teaching game design with the Game Specialized School, Chung-Ang University, in 2006. From 2009 to 2013, he served as a Professor with the Chungkang College of Cultural Industries. From 2009 to 2014, he has been the CEO of Nextgames, where he also served as the Project Leader. From 2015 to 2018, he was the Former Vice President of the Korea Game Developer Associations. He is currently an Assistant Professor with the Graduate School of Game, Gachon University, while serving as the Director of the Start-Up Education Center. He has many publications in various researching journals and books. His research interests include IT specifically on the techniques of computer games, AI, virtual reality technology, and interactive technology. He has been serving as the Editor-In-Chief for the Korea Computer Game Association, since 2016.