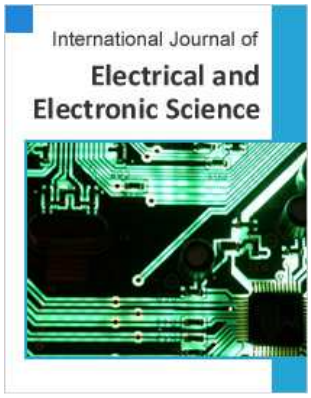
International Journal of Electrical and Electronic Science
2015; 2(3): 95-101
Published online October 20, 2015 (http://www.aascit.org/journal/ijees)
ISSN: 2375-2998

American Association for Science and Technology



Keywords
Web Based,
Advanced Intelligent Systems,
Application of Web Services for Artificial Intelligence,
Application Programming Interface

Received: August24, 2015
Revised: September29, 2015
Accepted: October1, 2015

Web Based Advanced Intelligent Systems

Rustom Mamlook*, Omer Fraz Khan

Department of Electrical & Computer Engineering, College of Engineering, Dhofar University, Salalah, Sultanate of Oman

Email address
rustom@du.edu.om (R. Mamlook)

Citation
Rustom Mamlook, Omer Fraz Khan. Web Based Advanced Intelligent Systems. *International Journal of Electrical and Electronic Science*. Vol. 2, No. 3, 2015, pp. 95-101.

Abstract
Provision of data to intelligent systems, for decision making, by making use of World Wide Web is one of the most promising areas of research in Artificial Intelligence. When a task is given to an intelligent system, its solution is unknown. It is the job of Artificial Intelligent (AI) systems to find the best solution based on the information available to it. With development in web technology and advent of dynamic web services the data is processed and distributed through web services. Such data is a valuable source of information for the proposed AI system. Web services are dynamic as parameters are passed on to its functions. Values returned from these web services (functions) can provide input for Intelligent Systems to accomplish various tasks. Our focus is on existing web services to be used as data source for the intelligent system. Web Services are considered to be ability provider to the system. We need to categorize web services which enable a particular ability for our intelligent prototype system. In our paper we have proposed a method to sort and categorize various forms of Web services in to areas and levels of expertise. We derived a technique for our Web Intelligent System to achieve tasks by utilizing one or more areas of expertise obtained from World Wide Web. Our AI-prototype's initial interaction with the World Wide Web is Application Programming Interface (API) directory through which it searches for an API related to the ability desired by the prototype, programmatically handshaking for data exchange protocols and initiating a flow of query and responses over transmission control protocol (tcp), user datagram protocol (udp) and Hypertext transfer Protocol (http/https).

1. Introduction

The web services are arranged with their index in to matrices called ability matrices. The approach we use is of organizing the input and output data returned from the web service in the form of matrices identified as result matrices. This gives us ability to analyze each web services data quality and make comparisons with data from other similar web services and finally feed the derived information to intelligent systems in the form of informative arrays.

Web Services are functions which follow the following basic principles:

1. Services are a set of functions which can be used by any client without requiring any change to the code of the service.
2. Such functions are accessible anytime with redundancy and failover mechanisms handled.

Before utilizing any of the web services for AI purpose we need to make sure that the automation is achievable in the following four primary functions.

a. Dynamic Discovery of web service.[1]

b. Service Integration: Compatibility of Interfaces (compatibility of exchanged parameters) from Web Service to that of AI Systems
c. Process Integration
d. Process Control

Universal Description and Discovery Integration (UDDI) directory helps in discovery of web service and Web Service Definition Language (WSDL) [2] serve documents to intelligent clients (algorithms) which use these documents to access the metadata or service tag for the set of functions provided by the web services. The extracted metadata and service discovery address will be saved in a cache as a repository for indexing purpose. Web services without sufficient metadata on their ability information are to undergo screening by intelligent algorithms which determine their ability matrix for categorization.

In this research we show a web ability extractor algorithm for a particular set of web services related to airline agent. We discuss a way in which a web service ability matrix can be composed by using service composition [3] and cached for usage by AI systems through remote access. Already available UDDI [4] discovery mechanisms were utilized for searching through a set of web service's WSDL documents and determine the descriptive tags determining the type and purpose of the data returned as Simple Object Access Protocol (SOAP) [5].

2. History & the Algorithm

Dynamic discovery of web services includes searching through the UDDI or API Directory. Once desired set of web services are dynamically discovered, selection of the right web service for the right task involves categorizing multiple web services according to their job description.

Work on automatic-composition algorithms for the Web Service Composition (WSC) [6] problem has been done in past such as Glaphplan, SATPlan and Integer Linear Programming (ILP).

We have also seen an approach of searching through web pages using matrices by Google's Page Rank algorithm by finding an eigenvector for an enormous sparse matrix [7].

Algorithm for development of an ability matrix relies on successful discovery and composition of web services. In our paper we discuss at least one method of:

a. ability matrix composition
b. ability matrix selection
c. results matrix derivation
d. application of result matrix to get the required output

3. Methodology

First step in our approach is to access the web service. It is done by authentication of the client (i.e. Our AI System) by a server hosting the web service. Handling the authentication procedure for Web Service using OpenID will allow web service crawler [8] to let the web services, present provisions for entering the key-value pair.

The Web Service publishes the expected parameters to be in the Response Message of OpenID. It also keeps a list of allowed OpenID Providers in service's WSDL.

- Simulator will programmatically provide the ID and the password (or a password callback handler) at the policy.xml (file type with data sorted in hierarchy).
- Discovery process of OpenID is done inside Apache Rampart [9] supplied by the simulator which then checks the discovered provider to be in a list of the published and trusted OpenID Providers, to allow the simulator to have privileges to use the web service.
- Only if the OpenID provider is a trusted Provider for handling the login procedure for the Web Service Provider, the Apache Rampart issues an Authentication Request Message along with the Parameters mentioned in the Web Service's WSDL and passes it to OpenID provider using "Direct Communication" along with the password of the simulator's registered ID, for authentication at the OpenID Provider
- After authentication is passed, OpenID provider responds to Apache Rampart with the Authentication Message.
- Apache Rampart wraps the Authentication Response Message to the OpenID Token and add it to the header of the Web Service Request SOAP Message
- Web Service Provider retrieves and verifies the Response Message.
- If the Verification is successful the Web Service is granted to the requester with returned results in JSON (JavaScript Object Notation) or XML (Extensible Markup Language, language designed to store transport data)format.
- JSON or XML parser splits results returned in to meaningful format readable by intelligent system.

Based upon the entry of input data to the Web Service's available functions from its API, a set of output data is received by the client. This set of data received comprises of key-value pairs that are saved in an output parameters matrix in a sequential manner. The parameters received are processed where data type and metadata is assigned to them

An ad hoc ability matrix is developed which consists of flags for execution. Each matrix sets or resets the flags generating a sequence in which the order of execution of a particular set of function in an API's should take place.

A quality metric matrix is formed which has the individual Boolean set for each API's successful execution and response. The quality of the result matrix depends upon the affirmation of set flags generated after the successful results from multiple API's.

The results returned by API's are stored in a result matrix. Each element of result matrix is another matrix consisting of the dimension of answers arranged as elements. Matrices are addressable and results are stored in a particular arrangement based on the ability matrix structure, predetermined by design.

During our pre-tests, algorithm was utilized with a soap based user Interface (UI) [9] is constructed. It simulated web

services for mission critical functions. Our algorithm used test cases to check the integrity of web service. Utilization of one web service also involved assertions and transfer of properties to the next web service such as cascading information retrieved from one web service calls to another.

Strength of the simulator depends mainly on its success in performing mission critical tasks. Results obtained by utilizing web services must be consistent with business logic, data integrity, exception handling and security. Algorithm is desired to be capable of

a. Comprehensively test a web service
b. Perform tests using realistic big data.
c. Determining the security level of the service.
d. Efficiently utilizing Intelligent Client and ability tester algorithms.
e. Tacking the test results obtained.
f. Testing web service under the effect of predetermined overload
g. Governing web service usage for various tasks.

Call-back approach is also used in testing web services where a query is sent to multiple web service and response returned is captured and saved. In Call-back approach parameterized multiple queries are passed as arguments to each function with in a web service. The results returned as a result of successful queries are composed in a result matrix. Functions which reject the query producing a failed response do not participate in generating the ability matrix while the ones which return the response are qualified candidates as active web service functions participating in building a particular ability which we require for our AI system's matrix. In this way we qualify some functions in a web service as functions contributing in composing ability matrix. Using this approach we utilize partial functionality of web service instead of disqualifying the web service as a whole.

In a test case scenario for utilizing multiple web services for composing an ability matrix, we have chosen the following set of web services [10]:

a. Weather Service
b. Currency Rates Service
c. Geo-IP Service
d. Time Service
e. Stock Price Ticker Service
f. Yearly Calendar Service
g. US Address Verification
h. Barcode Generator
i. North American Industry Classification System
j. United Nations Standard Products and Services Code
k. Medi Care Supplies
l. FedACH
m. FedWire
n. USA Weather Forecast
o. Mortgage Index
p. SunSetRiseService
q. Parcel Tracking Service

For-example, we propose a simulator which determines the time and observes the weather. Simulator effectively knows where it is at by using information returned from the weather web services [11]. Simulator was not provided the information by which it can search the required web service to achieve the task. It has to find the correct web service or the function of this web service, the correct number and data types of parameters to pass as an argument and the selection of correct vales from the returned dataset. In short, the simulator has to find the web service and determine the function of the web service.

Web services are segregated and arranged as two sets.

Web Services Indexes, Set 1: $\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$

Web Services Indexes, Set 2: $\begin{bmatrix} j & k & l \\ m & n & o \\ p & q & r \end{bmatrix}$

Success of the model will rely upon the successful derivation of ability matrix from web services and the quality [12] of the respective results matrix.

The respective function set matrix is arranged as one dimensional linear vector corresponding to each web service. Each matrix consists of all set of functions provided by each Web Service Index matrix element. Size of each function set matrix depends upon the number of public functions discovered through UDDI service for a particular web service set.

a = [af1, af2, af3, af4, af5.....]
b = [bf1, bf2, bf3, bf4, bf5....]
c = [cf1, cf2, cf3....]
d = [df1, df2, df3, df4, df5...]
e = [ef1, ef2, ef3, ef4....]
f = [ff1, ff2, ff3, ff4....]
g = [gf1, gf2, gf3, gf4....]
h = [hf1, hf2, hf3, hf4....]
i = [if1, if2, if3, if4....]
j = [jf1, jf2, jf3, jf4....]
k = [kf1, kf2, kf3, kf4....]
l = [lf1, lf2, lf3, lf4....]
m = [mf1, mf2, mf3, mf4....]
n = [nf1, nf2, nf3, nf4....]
o = [of1, of2, of3, of4....]
p = [pf1, pf2, pf3, pf4....]
q = [qf1, qf2, qf3, qf4....]
r = [rf1, rf2, rf3, rf4....]

Each function is defined in terms of input parameters and output returned values. The Data Type and the input parameters id are saved in input parameters matrices. Data Types of keys are arranged in a matrix called xfnKeyDt. The Key and Value pair is arranged in matrices xfnKi and xfnVi, respectively. (Where x is a lower case alphabet denoting web service index and n is an integer denoting the function's index).

af1KiDt = [int, int, int, int] (data type of each input key is int)
af1Ki = [a, b, c, d] (key)
af1Vi = [?,?,?,?] (value)

"?" will be replaced by test values (w,x,y,z) from test sequence matrix.

Each value passed in as argument to af1Val matrix has a response from the function which is the returned key value pair. We define the returned value in three matrices as follows.

af1KoDt = [int...] (Each function may return more than one key, during our study we assume each function returns (outputs) only one key and matrix af1KoDt hold the data types of returned key)

af1Ko = [?. ...] (name of returned key as defined by definition of the function 'f1' related to web service 'a')

af1Vr = [r(abcd), r(bacd), r(bcad)....] (The number of value returned will be equal to the number of permutations of input parameters)

We send the parameters to all the functions of the web service, catch the exception using the underlining catch-exception method of the web service providers. An exception raised is considered as the inability of web service function to meet one of the ability matrix minimum criteria by either not returning the results or returning values which have no value to the ability matrix.

Sending the parameters is done in a way as to exhaust all possible sequences in which the parameters will be provided to the input of the functions and the respective answers (permutations) returned from each sequence will be saved in corresponding result matrices. Each input matrix will have its corresponding results matrix. Each element in the result matrix (raf1) will be compared to its corresponding element in the input matrix.

raf = [af1KoDt, af1Ko, af1Vr,
af2KoDt, af2Ko, af2Vr,
af3KoDt, af3Ko, af3Vr,
af4KoDt, af4Ko, af4Vr,
af5KoDt, af5Ko, af5Vr,
. ....., ..., ...]

The above matrix is 3 times the size of function set matrix as we saved the returned key's data type, key's identity and key value.

The matrices that were returned as a result of permutations of input parameters further created matrices that were equal to the number of permutations of input parameters E.g. in case of permutation (a,b,c,d), n number of results are returned.

raf1(permt(abcd)) = [af1KoDt, af1Ko, af1Vr1,
af1KoDt, af1Ko, af1Vr2,
af1KoDt, af1Ko, af1Vr3,
af1KoDt, af1Ko, af1Vr4,
af1KoDt, af1Ko, af1Vr5,
. ....., ..., ...
af1KoDt, af1Ko, af1Vr n]

Results matrix is a parse matrix with '0' as no value returned while a value returned is valid only if the data type of the desired result (afnVr n) matches with the data type (afnKoDt) of returned value (afnVr n), where n = permutations as an integer.

The knowledge about metadata on returned value is extracted from the SOAP documents.

Results matrix, "R" has information about the query results corresponding to the multiple ways in which the function was challenged on providing the answer.

Derivation of result matrices Ra, Rb... and Rx gives us the following:

Ra = [af1Vr1, af1Vr2, af1Vr3, af1Vr4, af1Vr5,
af2Vr1, af2Vr2, af2Vr3, af2Vr4, af2Vr5,
. ....., ..., ..., ..., ...,
afnVr1, afnVr n, afnVrn, afnVrn, afnVrn]
Rb = [bf1Vr1, bf1Vr2, bf1Vr3, bf1Vr4, bf1Vr5,
bf2Vr1, bf2Vr2, bf2Vr3, bf2Vr4, bf2Vr5,
. ....., ..., ..., ..., ...,
bfnVrn, bfnVrn, bfnVrn, bfnVrn, bfnVrn]
R... = [...f1Vr1,. ..f1Vr2. ..f1Vr3,. ..f1Vr4,. ..f1Vr5,
. ..f2Vr1,. ..f2Vr2,. ..f2Vr3,. ..f2Vr4,. ..f2Vr5,
. ....., ..., ..., ..., ...,
. ..fnVrn,. ..fnVrn,. ..fnVrn,. ..fnVrn,. ..fnVrn]
Rx = [xfnVr1, xfnVr2, xfnVr3, xfnVr4, xfnVr5,
xfnVr1, xfnVr2, xfnVr3, xfnVr4, xfnVr5,
. ....., ..., ..., ..., ...,
xfnVrn, xfnVr n, xfnVrn, xfnVrn, xfnVrn]

If result set is finite and denoted by R, we can write result matrices separately for each function as follows:

R(f1) = [af1Vr1,af1Vr2,af1Vr3,...., af1Vr n]
R(f2) = [af2Vr1,af2Vr2,af2Vr3,.....,af2Vrn]
R(fn)=[afnVr(n), afnVr (n+1), afnVr(n+2),. ....., afnVr(n+p)]

Where p is the maximum number of permutations of input variables.

Informative matrix which is to be fed into the AI-System is the final result Matrix (R) of the following form:

R = [R (f1) + R (f2) + R (f3) +. .... + R (fn)]

Similarly, we can derive the value's Data-Type Matrix and key's Identity Matrix in a key value pair. This constitutes our metadata corresponding to results, hence

RDt = [R (af1KoDt) + R (af2KoDt) +...+ R (afnKoDt)]
RKo = [R (af1Ko) + R (af2Ko) +. ... + R (afnKo)]

Next step is to design an algorithm which can find the quality of R and is able to extract each elements identity and data type from the matrices RKo and RDt respectively.

Quality of R is determined by the instances of required data found in the result matrix. If the term is found in set of Keys returned (Ko) and the Keys' Data type matches with the data needed to be consumed by AI, then we can further do an algorithmic analysis of the consistency of data by matching it with further results (R1,R2...Rn) obtained from additional web services search.

Web Services can also be linearly arranged into an initial index matrix as shown below:

Web Services- Initial Index Matrix
ws [ws1, ws2, ws3, ws4, ws5]

For each web service matrix, multiple ability matrices were derived. Generation of each ability matrix involved the selection of those methods (operation contracts) which are related to the ability. The arrays below shows the subset functional entities that is usually returned by the UDDI service attached with the web service and found in proxy class derived from the web service.

Ability Matrices- can be unbalanced because web services were observed to have varying number of functions. In matrix

below ws1 (web service # 1) is the identifier of web service and its related functions are ws1f1 and ws1f2. In test case scenario we assumed matrices to be balanced i.e. each matrix is having two functions.

ws1 [ws1f1, ws1f2]
ws2 [ws2f1, ws2f2]
ws3 [ws3f1, ws3f2]
ws4 [ws4f1, ws4f2]
ws5 [ws5f1, ws5f2]

After investigating each function of a web service the identity of input and output parameters and data-type matrices are derived, e.g. the following function ws1f1 of web service (ws) from set 1 (ws1) requires city, country and unit of temperature in its own defined sequence, whereas function ws2f1 of web service (ws) from set 2 (ws2) requires same set of parameters but in a different sequence from ws1f1.

ws1
ws1f1 [string city, string country, string unit]
return temp;
ws1f2 [string city]
return code;
ws2
ws2f1 [string country, string city, string unit]
return temp;
ws2f2 [string city]
return code;

The aforementioned passing of parameters to get information returned (output) from functions is handled by sequence generator using permutations of input parameters.

Output (returned) parameter's Data-Type Matrices are generated when each function returns a value in response to the passed parameters in a particular sequence. Ideally a response matrix rws1should have only one non-null value as response matching with only one sequence of input parameter out of multiple sequences entered but there is a possibility that many elements in each Data Type matrix may hold non null values.

rws1 [rws1f1 rws1f2]
rws2 [rws2f1 rws2f2]
rws3 [rws3f1 rws3f2]
rws4 [rws4f1 rws4f2]

We used test input parameters Data-Type matrices-Data type having mixed data types of input parameters to test for their resulting output.

tf1 [string, int, long]
tf2 [string, string, string]
tf3 [long, long, char]
tf4 [char[ ], int[ ], string]
tf5 [*string, int, int]
tf6 [int, int, int]
tf7 [double, double, double]

Testing Parameters with all Sequences Matrices - fixed based on parameters. The matrices will pass the actual values to input parameters in varying sequences.

tf1 [salalah, C, Oman]
tf2 [Oman, C, salalah]

Test return matrices – Actual values returned as by the function in response to the passed sequence of parameters.

tr [trf1,trf2,trf3,trf4,trf5,trf6,trf7]

By using this approach we obtained results from multiple web services for one particular city in response to a query for its weather. The response matrices also known as the result matrix was derived. It was beneficial in keeping multiple values from different sources of web services for allowing the smart system to make its decision on the selection of quality values.

4. Pseudo Code

a. Input query Parameters
b. Compose "WS" Matrix as a set of addresses of web services "WSn" related to Input query.
c. Generate a "WSF" functions matrix with each element pointing to other sub-Matrix "WSnF".
d. Fill WSnF matrix with all possible public function's addresses "WSnFn" returned by running batch UDDI discovery on WSn.
e. Collect the results received in the form of Key-Value pair for result returned by "WSnFn" for each sequence of input parameters (permutations).
f. Collect the Data-Type "RDt" of returned values and save them in matrix "RDt".
g. Compare values in results with values satisfying the required results from initial query.

The Flowchart of our methodology is presented in Figure 1.

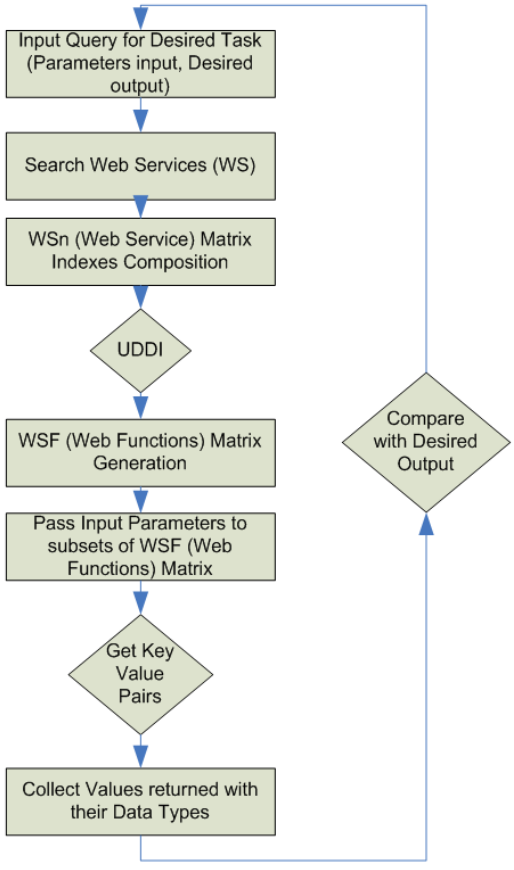


Figure 1. *Flowchart of our methodology.*

5. Case Study

An intelligent appliance control was developed at Engineering Labs at Dhofar University using the method described in our paper. Year round weather data was provided by different web services out of which a few services provided real time data corresponding to the actual on site measurements. We were successful in differentiate between the quality of data that was available to us and utilize it for temperature calculations in one of the applications. During the course of our case study we were able to explore ways in which web services are located and indexed using UDDI Discovery [13] mechanism into a Web Service composer. We were able to implement a WSDL parser or a WSDL reader for determining information about input and output parameters. We used SOAP (JSON and XML) protocols and available SOAP parsers to obtain metadata such as Data Type and Key, Value Pairs in matrix form.

We were able to implement a C# algorithm of a Web Services batch caller in which we can engage multiple web service providers to provide us with WSDL documents on each service. We managed to make a consumer of web service in the form of a simulator with ability to take user query input and display the number of results obtained matching the query.

Our main focus was to create ability matrix generator and indexer which not only downloads answers to queries in the form of matrices but also has the ability to form a matrix cache of results along with the data Type and naming scheme used by the web service functions to define the answers.

6. Results

A permutation engine was designed to generate sequence of inputs from user entered queries and feed them as test input parameters to each function of web services.

Results Matrix Generator from results obtained from permutation engine's output and Cache them in addressable memory along with ability to locate each result's metadata from a metadata matrix generator.

Informative Array formed of Results + metadata are fed into the simulator and displayed to the user through GUI.

7. Conclusion

Each matrix has the ability to expand and contract (learn and relearn) based on the availability of results returned by web service and its functional ecosystem. The probability of correct results is better with the expansion of the number and quality of web services. The association of function's input and output parameters into an ability matrix holds a striking analogy to the faculties of a human Brain. For example hypothalamus performs some functions of human activities and similarly one ability matrix will be able to perform particular functions until we develop another ability matrix to perform other function. Web provide the necessary inputs to the ability matrix similar to the vision, sound, sense of touch provides the ability to the human brain.

We have seen our approach to work on known web services in which parameters input and returned parameters are expected to be of simple known types. Semantic Web [14, 15] is an area which needs further exploration and integration while working in realms where little or no information is available about vast amount of data available over World Wide Web.

Acknowledgment

The research council (TRC) is the premier technical body in Oman. As its representatives are in the campus of Dhofar University in Oman, it is the best suitable organization for faculties like us. Furthermore the peoples in the TRC are helpful and innovation friendly.

Internet Service Provider, Oman telecommunication authority (Omantel) has a vast infrastructure of fiber optic wired network as well as wireless data connectivity. It is due to their outstanding provision of 24/7 data connectivity that has made our system successful with uninterrupted communication channel available at our disposal round the clock.

We are grateful for the research labs and resources provided by the Dhofar University Salalah, Oman.

References

[1] D. Richards, M. Sabou, S. van Splunter, F.M.T. Brazier. Artificial Intelligence: a Promised Land for Web Services, retrieved from http://comp.mq.edu.au/~richards/, last updated on July, 2015.

[2] AmnaEleyan, Liping Zhao, Extending WSDL and UDDI with Quality Service Selection Criteria, 2010.

[3] Seog-Chan Oh, Dongwon Lee, and Soundar R.T. Kumara, A Comparative Illustration of AI Planning-based Web Services Composition, 2005.

[4] UDDI discovery mechanisms retrieved from http://uddi.xml.org last updated on September 2015.

[5] Brian Suda, SOAP Web Services: Publications at University of Edinburgh 2003.

[6] OWS 2 Common Architecture: WSDL SOAP UDDI, Open Geospatial Consortium Inc, 2004.

[7] A project for answering queries to the typed input questions available at http://webknox.com last update in Sept, 2015.

[8] Kurt Bryan and Tanya Leise, The $25,000,000,000* Eigenvector the Linear Algebra Behind Google, 2006.

[9] An Open Source web crawler project available at http://nutch.apache.org/, last updated in 2014.

[10] Web Services available at the following: http://www.webservicex.net/ws/WSDetails.aspx?CATID=12&WSID=56, last updated in 2015.

[11] Weather providing Web Service retrieved from http://wsf.cdyne.com/WeatherWS/Weather.asmx in Aug, 2015.

[12] Web Service Search Project retrieved from http://db.cs.washington.edu/webService/woogle_vldb.pdf, last retrieved in 2014.

[13] SOAP UI Web Service functional Tests at http://www.soapui.org/Getting-Started/web-service-sample-project/All-Pages.html, last updated in 2015.

[14] Katia Sycara, Dynamic Discovery, Invocation and Composition of Semantic Web Services, 2004.

[15] Page 8. Autonomous Semantic Web Service Discovery, Methods and Applications of Artificial Intelligence: Book on Third Helenic Conference on AI, SETN 2004, Samos, Greece, May 5-8, 2004, Proceedings. Volume 3, Springer. George A. Vouros, ThemistoklisPanayiotopoulo, Pages: 546.