Research Journal of Applied Sciences, Engineering and Technology 10(5): 562-569, 2015
DOI:10.19026/rjaset.10.2464
ISSN: 2040-7459; e-ISSN: 2040-7467
© 2015 Maxwell Scientific Publication Corp.
Submitted: November 27, 2014 Accepted: January 8, 2015 Published: June 15, 2015

Research Article

Rule Based Fine Tuned Web Service Discovery using JESS

[1]A. Gnanasekar and [2]R.M. Suresh
[1]Department of Computer Science and Engineering, R.M.D. Engineering College,
[2]Sri Muthukumaran Institute of Technology, Anna University, Chennai, India

Abstract: The aim of this study is to present a novel approach which uses Java Expert System Shell (JESS) to intelligently infer the best and most relevant Web service. Since the Web Service has developed as a service provider in all areas, the Service discovery has become indispensable as Web Service discovery algorithms return more than one Web Service with same functionalities. Generally, the functional properties of the Web Services, such as Input, Output, Precondition, Effects (IOPE) are considered in the composition phase whereas the non-functional properties, namely Quality of Service (QoS) parameters are used in selecting the most appropriate Web Service. User Rule-based searching is accomplished along with users' preferences which map the rules of the user profile into JESS script to discover the desired Web Service. As a result of our research we support developers to combine and implement the Web Service description and discovery approaches that will ensure best and most relevant service and performance of service oriented architecture.

Keywords: Java expert system shell, quality of service parameters, service oriented architecture, web service discovery

INTRODUCTION

Service discovery is part of the service-oriented architecture and the current approaches for Web Services discovery is by providing semantic layer on Web Service standard components such as WSDL (Erik *et al*., 2001) and UDDI registry (Luc *et al*., 2004), which only supported keyword search. Web Service discovery becomes semantic search which alleviates the limitations in keyword search values in UDDI. For Semantic Web Service annotation, OWL-S coalition has promoted Ontology Web Languages for Services (OWL-S) (David *et al*., 2004), with rich semantic annotations. There are many works which have been proposed towards the Semantic web Service discovery. In this study the rule based inference engine namely JESS (Friedman-Hill, 2007) is effectively used along with the user profile of the customer for service discovery. A large number of rule engines are available as open source software. Some of the most popular engines include JESS, Algernon, Sweet Rules and Bossam. We chose JESS a forward-chaining rule engine, as the rule engine for the service discovery based on the QoS. JESS works seamlessly with Java and is very easy to use and configure. Each Jess rule engine holds a collection of knowledge nuggets called facts. Every fact has a template. The template has a name and a set of slots and each fact gets these things from its template (Friedman-Hill, 2007).

LITERATURE REVIEW

Recently, QoS-aware Web service Discovery is an active research issue both in industry and academia that attracts a lot of researchers. Many studies have been carried out and several approaches have been proposed for this problem. Yang *et al*. (2008) proposed Java Expert System Shell (JESS)-enabled context elicitation system featuring an ontology-based context model that formally describes and acquires contextual information pertaining to service requesters and Web services. Based on the context elicitation system, we present a context-aware services-oriented architecture for providing context-aware Web service request, publication and discovery. Gunasri and Kanagaraj (2014) proposed a Semantic web service discovery framework for finding semantic web services by making use of natural language processing techniques and clustering method. By make use of natural language processing used keyword matching with context of service description. It has accurate matching because Word net gives an exact sense for a particular web service domain and cluster Terms we can improve the optimization and eliminating irrelevant services and gives accurate service discovery. Aklouf and Rezig (2009) proposed approach exploits expert systems that aim at adding new functionalities to Web services according to their rule-base defined by the knowledge engineer or the system administrator using an ontology.

Corresponding Author: A. Gnanasekar, Department of Computer Science and Engineering, R.M.D. Engineering College, Anna University, Chennai, India

Kritikos and Plexousakis (2009) present a QoS in the context of WSs. Its main contribution is the analysis of the requirements for a semantically rich QoS-based WSDM and an accurate, effective QoS-based WS Discovery (WSDi) process. In addition, a road map of extending current WS standard technologies for realizing semantic, functional and QoS-based WSDi. Finding an appropriate and best service from a group of services with the same functionality is a service selection challenge (Liangzhao *et al*., 2004; Maximilien and Singh, 2004; Mukhopadhyay and Chougule, 2012; Nair and Gopalakrishna, 2010). Therefore, besides the functionalities of web services, nonfunctional attributes (QoS) are also important in service discovery. In order to meet users' functional and non-functional requirements in service discovery, many QoS-based methods have been proposed (Harshavardhanan *et al*., 2012; Torres *et al*., 2011).

Gouscos *et al*. (2003) classifies QoS attributes as a static and a dynamic group. For example, price, promised response time and probability of failure are static attributes stored in UDDI. On the other hand, actual response time and failure rate are stored in WSDL, or provided by an information broker. This method is quite easy and straightforward, but cannot solve the problem caused by obsolete and out-of-date QoS information. Ran (2003) proposes an extended UDDI model which involves a new role, namely a web service QoS Certifier, in a traditional SOA model. It consists of three roles (service providers, service requesters and UDDI registry). This verifies whether the quality of a service is the same as the service provider promised before registering the service in UDDI. Thus, a service consumer can first issue a query to the web service QoS Certifier to verify the QoS of a target service. However, this does not provide a reliable algorithm to validate the credibility of claimed QoS. It is unable to assure the correctness of real QoS when web services are changed or updated in the future. Huang *et al*. (2009) proposes a three-stage service selection scheme based on different types of QoS such as functional matchmaking, text-based QOS matchmaking and numeric-based QoS matchmaking. In a text-based QoS matchmaking stage, keyword search and service category search are provided by UDDI. There are two scenarios in a stage of numeric-based QoS matchmaking: single QoS-based service discovery, which selects the service with the best QoS attributes for users and QoS-based Optimization, which selects the service with the best performance in an entire workflow. Al-Masri and Mahmoud (2007) proposed a Web Service Repository Builder (WSRB) framework. In this framework, a Web Service Crawler Engine (WSCE) sends multiple queries to several UDDI registry centers, according to users' requests and then gets all the QoS results. The QoS of services with the same functionality will be stored in a matrix and all QoS attributes are then normalized. The ranking scores for each service will be computed by a weighted sum of the value of QoS attributes, where the weighting for each attribute is determined by a user according to his/her preferences. Lin *et al*. (2014) proposed a trustworthy two-phase service discovery algorithm based on collaborative filtering and QoS in order to recommend good services from the same functional service group to users. The recommendation process, can verify the correctness of QoS for web services. Therefore, the recommended services not only meet users' functional requirements, but also have correct QoS information. Liu *et al*. (2012) proposed Branch and Bound for Execution Plan Selection (BB4EPS) algorithm that creates a plan for service composition using the aggregated affect of the QoS attributes. The aggregation is studied for services connected in different structures/patterns. Both availability, reliability is studied separately and their combined effect is not evaluated. Liu *et al*. (2013) proposed a model the QoS-aware service composition problem as a conventional combinatorial optimization problem, we transform this problem to be a local optimization problem by decomposing global QoS constraints into a set of local constraints and design a new global QoS constraints decomposition model that used to find the optimal local QoS constraints combination for each service. In this study proposes a new service discovery approach based on Web service QoS knowledge using java expert.

SEMANTIC WEB SERVICE DISCOVERY USING JESS

However, in this study we propose a novel method which utilizes the potentiality of forward inference of the JESS. The process of the proposed method is depicted in Fig. 1. In one of our previous works (Gnanasekar and Suresh, 2014), we have used a discovery algorithm, which is applied to the OWL-S service retrieval test collection, OWL-S TC (version 4.0). For any particular query, this algorithm has returned more than one service with same functionality. For example, when the user wants to know the price of a book, the user had entered "Book" as input and "Price" as output. The algorithm has returned seven services with the same functionality which the user is desired.

However, it is obvious that the user may not be interested in all the seven Web Services, instead may be interested in one service with good QoS. It would be better to incorporate, the discovery algorithm with a fine tuned search engine to extract the relevant service the user expects. Hence, a user profile is created with the user's preferences. The user profile is converted into JESS facts, which are the rules for the JESS inference engine.

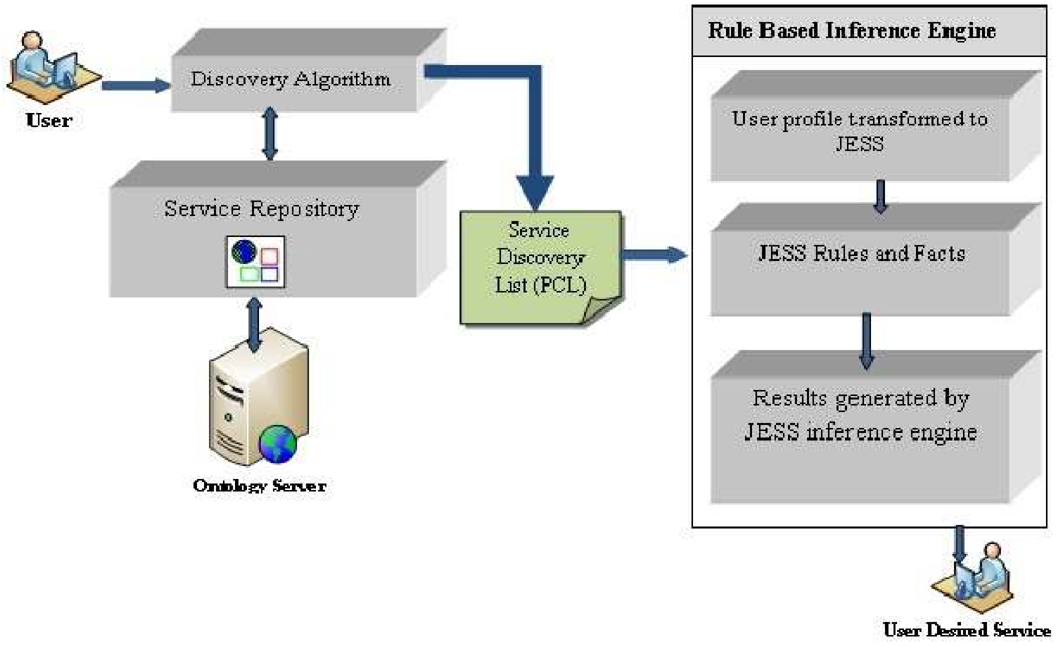

User
Discovery Algorithm
Service Repository
Ontology Server
Service Discovery List (PCL)
Rule Based Inference Engine
User profile transformed to JESS
JESS Rules and Facts
Results generated by JESS inference engine
User Desired Service

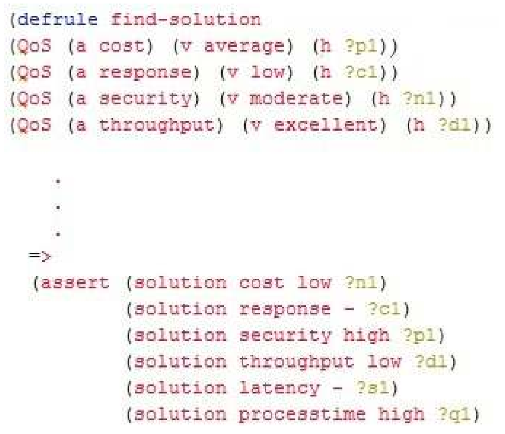
Fig. 1: Architecture of rule based service discovery based on user profile

Rule based inference engine: This module first converts the preferences in the user profile into JESS rules (Liangzhao *et al*., 2004). A JESS rule is something like an *if... then* statement in a procedural language, but it is not used in a procedural way. In the simplest terms, this means that JESS purpose it to continuously apply a set of rules to a set of data. We can define the rules that make up our own particular rule-based system. Jess rules look something like this:

```
(defrule welcome – toddlers
"Give a special greeting to young children"
(person {age < 31})
=>
(printout t "Hello, little one!" crlf))
```

This rule has two parts, separated by the "=>" symbol. The first part consists of the LHS pattern (person {age<3}). The second part consists of the RHS action. Each Jess rule engine holds a collection of knowledge nuggets called facts. Every fact has a template. The template has a name and a set of slots and each fact gets these things from its template.

Based on this hypothesis, the user profile preferences are converted into the JESS rules. The excerpts of one of such rules are shown in Fig. 2. For the fine tuned Web Service discovery, the JESS inference engine compare the user's rules drawn from the user profile registry and selects only the appropriate service information.



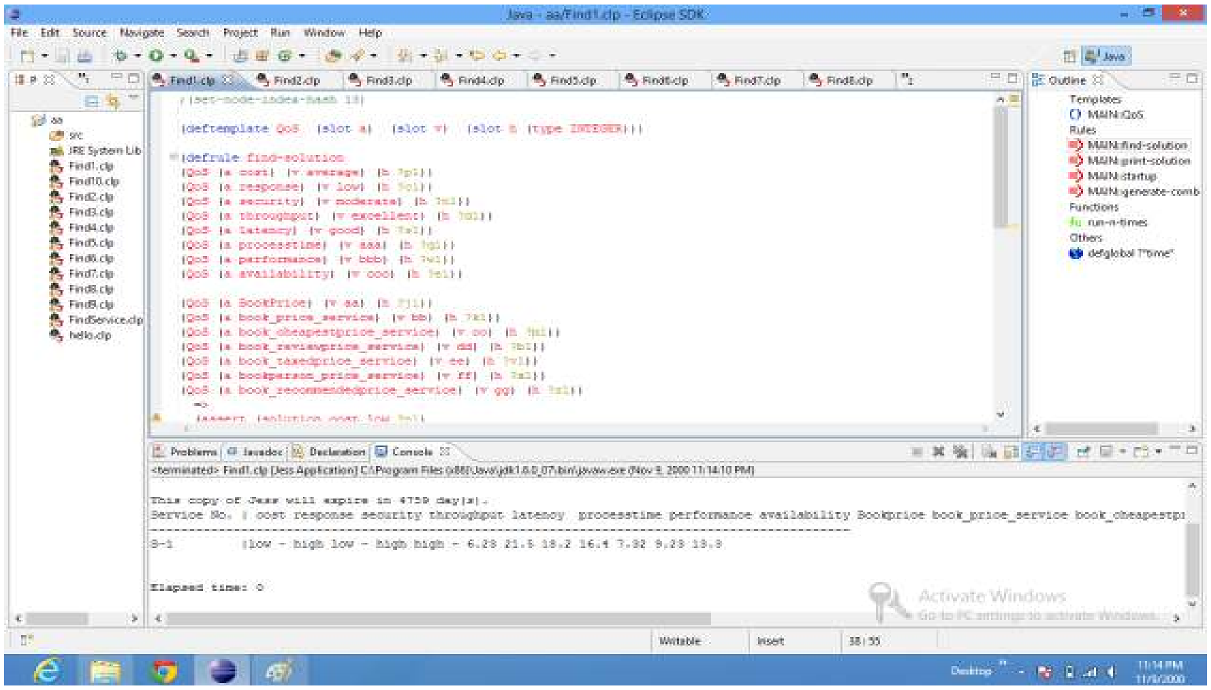
```
(defrule find-solution
(QoS (a cost) (v average) (h ?p1))
(QoS (a response) (v low) (h ?c1))
(QoS (a security) (v moderate) (h ?n1))
(QoS (a throughput) (v excellent) (h ?d1))

    .
    .
    .
  =>
  (assert (solution cost low ?n1)
          (solution response - ?c1)
          (solution security high ?p1)
          (solution throughput low ?d1)
          (solution latency - ?s1)
          (solution processtime high ?q1)
```

Fig. 2: Excerpts of rules of the preferences of the user profile

IMPLEMENTATION AND EXPERIMENTAL RESULTS

In this study, we use JESS 7.1 as the rules engine, because it can be integrated into Eclipse 3.5 as a plug-in with no extra development effort. We have used a computer with a 1.73 GHz Pentium Dual CPU and 1.50 GB of RAM was running Windows 7 OS, Java SDK 1.4.1 and JESS 7.1. For the implementation we have taken seven Web Services and eight QoS parameters such as Cost, Response, Security, Latency, Throughput, Process Time, Performance and Availability. The user profile is updated with these QoS preferences. This users profile with their preferences is first translated into JESS knowledge base. This knowledge base is then converted into JESS rules as shown in Fig. 3.



```
;(set-node-index-hash 13)

(deftemplate QoS  (slot a)  (slot v)  (slot b (type INTEGER)))

(defrule find-solution
(QoS (a cost) (v average) (b ?p1))
(QoS (a response) (v low) (b ?o1))
(QoS (a security) (v moderate) (b ?n1))
(QoS (a throughput) (v excellent) (b ?d1))
(QoS (a latency) (v good) (b ?x1))
(QoS (a processtime) (v aaa) (b ?g1))
(QoS (a performance) (v bbb) (b ?w1))
(QoS (a availability) (v ccc) (b ?e1))

(QoS (a BookPrice) (v aa) (b ?j1))
(QoS (a book_price_service) (v bb) (b ?k1))
(QoS (a book_cheapestprice_service) (v cc) (b ?m1))
(QoS (a book_reviewprice_service) (v dd) (b ?b1))
(QoS (a book_taxedprice_service) (v ee) (b ?v1))
(QoS (a bookperson_price_service) (v ff) (b ?z1))
(QoS (a book_recommendedprice_service) (v gg) (b ?z1))
  =>
  (assert (solution cost low ?n1))
```

```
<terminated> Find1.clp [Jess Application] C:\Program Files (x86)\Java\jdk1.6.0_07\bin\javaw.exe (Nov 9, 2000 11:14:10 PM)

This copy of Jess will expire in 4759 day(s).
Service No. | cost response security throughput latency  processtime performance availability Bookprice book_price_service book_cheapestpr
------------------------------------------------------------------------------------------------
S-1         |low - high low - high high - 6.23 21.5 18.2 16.4 7.32 9.23 13.3


Elapsed time: 0
```

Fig. 3: Screen shot for the JESS which inferred probabilities of user given QoS options for book purchase example

Table 1: JESS which inferred probabilities of different services for book purchase service here, H = high, L = low and M = medium

Test case	Cost	Response	Security	Latency	Throughput	Process time	Performance	Availability
1	L	-	H	L	-	H	H	-
2	L	H	H	L	H	H	H	M
3	M	H	M	L	H	M	H	H
4	L	H	-	M	-	-	-	M
5	-	-	-	-	H	H	H	H
6	-	H	H	-	-	L	H	-
7	L	-	-	-	-	M	M	H
8	L	-	-	L	H	-	H	H
9	-	-	H	L	-	-	H	H
10	M	M	-	M	H	-	M	-

Test case	Book price	Book_price_ service	Book_cheapest price_service	Book_reviewprice _service	Book_taxed price_service	Book person_ price_service	Book_recommended price_service
1	6.23	21.5	18.20	16.4	7.32	9.23	13.3
2	8.19	20.2	17.60	19.2	6.32	10.20	12.4
3	5.78	21.1	16.60	17.9	4.55	11.40	18.6
4	8.76	17.3	14.20	20.1	9.13	11.20	17.2
5	8.82	19.4	17.40	16.3	5.24	12.60	18.2
6	11.20	17.6	11.30	17.2	7.28	13.90	16.9
7	12.20	16.4	17.20	14.6	8.33	15.30	13.2
8	5.83	20.6	18.30	16.6	5.23	9.81	18.7
9	7.10	18.8	16.80	17.3	7.73	10.40	17.2
10	10.20	18.4	8.89	17.5	14.30	13.70	16.8

Suppose a user wants to select a best service with high security, performance and processing time and low latency and cost. In this case, the probability of book_price_service is 21.5% followed by the probability achieved by book_cheapestprice_service with 18.2%. That is based on the probability distribution of various dependency convergences; the inference engine of JESS inferred that for the given preference book_price_service is a better choice than any other services. Here the user has ignored or is not interested in the remaining QoS such as response, availability and throughput. The use of the JESS has reduced processing time and facilitated the Web Service discovery based on the customer satisfaction. The powerful inference engine of JESS infers the facts based on the rules and produced the outputs as show in Table 1.

Semantic web service discovery using belief network:

Belief network: A belief network is also called Bayesian network, is a graphical representation of a probabilistic dependency model. It consists of a set of nodes, where each node represents stochastic variables

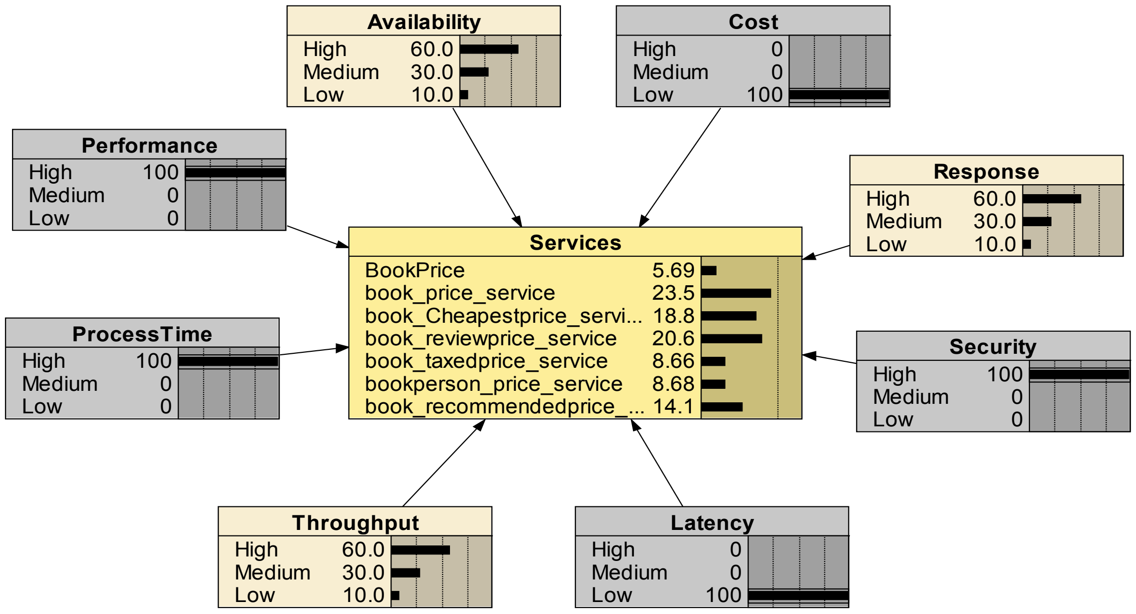


Fig. 4: A compiled belief net shows the probabilities of user given QoS options for book purchase example

and interconnecting arcs represent the causal influences between these variables. It is the use of Bayesian calculus to determine the state probabilities of each node from the predetermined conditional and prior probabilities that distinguishes Bayesian belief networks from other probabilistic dependency models.

Implementation and experimental results: Each node in the belief net must have a relation stored at each node, which expresses the value of that node in terms of its parents (or as a constant if the node has no parents). The node may be deterministic or probabilistic. If the node is probabilistic, then the relation must provide a probability for each state of the child, for each possible configuration of parent values.

In book purchase example, there are seven services from OWL-S TC, such as Book Price, book_price_service, book_cheapestprice_service, book_reviewprice_service, book_taxedprice_service, bookperson_price_service, book_recommendedprice_service. Based on this hypothesis, a belief net for the QoS parameters such as availability, cost, response time, security, latency, throughput, process time and performance is constructed for the Book Purchase example. However, the user may be in trouble selecting the best service among these services. Suppose a user wants to select a best service with high security, performance and processing time and low latency and cost. The probabilities of the services are inferred by the Belief Net is as shown in Fig. 4. In this case, the probability of book_price_service is 23.5% followed by the probability achieved by book_reviewprice_service with 20.6%. That is based on the probability distribution of various dependency convergences; the belief net inferred that for the given preference book_price_service is a better choice than any other services. Here the user has ignored or is not interested in the remaining QoS such as response, availability and throughput. The various possible probabilities of the Book Purchase example are given in Table 2.

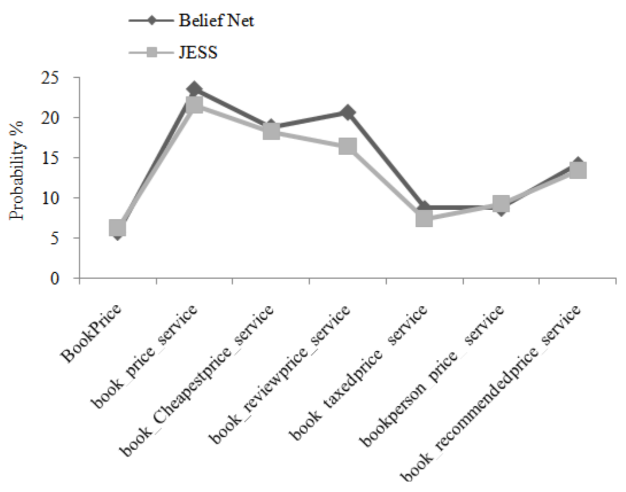
DISCUSSION

In Belief network is to be noticed that, for each different requirement, the probability of the services differ according to the probability distribution given to the QoS parameters. For the Book Purchase example book_price_service is having a highest probability than any other services except test case numbers 6, 7 and 10. In the test cases 6 and 7, book_recommendedprice_service is having the probability of 18.9 and 20.5%, respectively, whereas in test case number 10, it is book_reviewprice_service which attained the probability of 21.5%. It is to be noticed that in JESS, book_price_service is having a highest probability than any other services except test case numbers 4 and 7. In the test cases 4 and 7, book_reviewprice_service is having the probability of 20.1% and book_cheapestprice_service is having the probability of 17.2%. From the Table 1 and 2, it is found that book_price_service which would be better choice than any other service. Figure 5 shows the performance of Belief Network and JESS. From the experimental result, it is found that belief network to intelligently infer the best and most relevant service compare to the JESS.

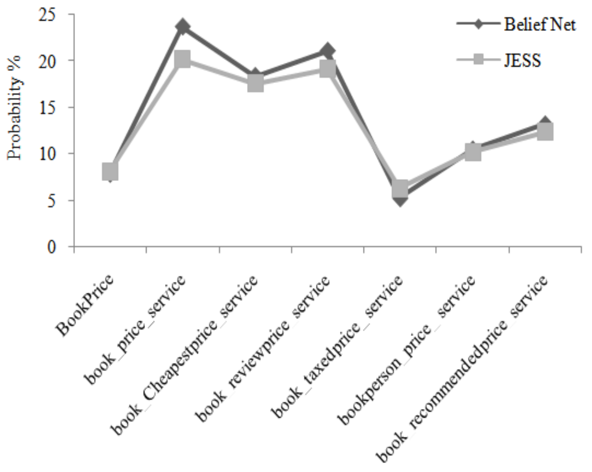
Table 2: Probabilities of different services for book purchase service

Test case	Cost	Response	Security	Latency	Throughput	Process time	Performance	Availability
1	L	-	H	L	-	H	H	-
2	L	H	H	L	H	H	H	M
3	M	H	M	L	H	M	H	H
4	L	H	-	M	-	-	-	M
5	-	-	-	-	H	H	H	H
6	-	H	H	-	-	L	H	-
7	L	-	-	-	-	M	M	H
8	L	-	-	L	H	-	H	H
9	-	-	H	L	-	-	H	H
10	M	M	-	M	H	-	M	-

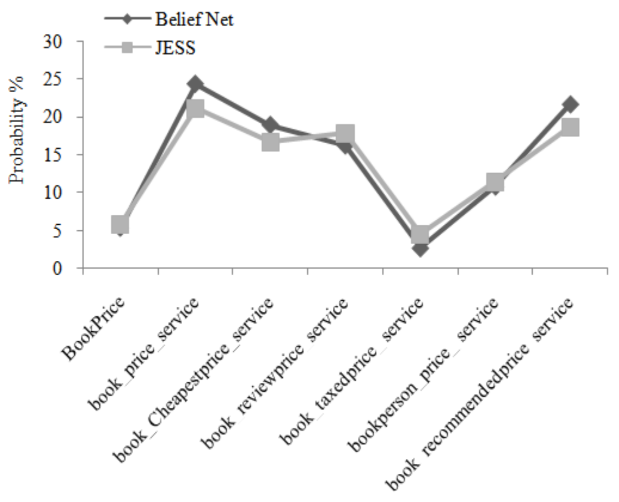
Test case	Book price	Book_price_ service	Book_cheapest price_service	Book_review price_service	book_taxed price_ service	Book person _price_ service	Book_ recommended price_service
1	5.69	23.5	18.80	20.6	8.66	8.68	14.1
2	7.89	23.7	18.40	21.1	5.26	10.50	13.2
3	5.41	24.3	18.90	16.2	2.70	10.80	21.6
4	9.88	21.4	15.10	16.1	8.23	10.60	18.7
5	7.80	22.9	17.50	16.7	3.64	11.60	19.9
6	12.20	17.6	12.20	17.1	8.58	13.40	18.9
7	14.00	16.3	10.90	15.7	8.30	14.20	20.5
8	5.92	24.2	18.80	17.2	3.23	10.80	19.9
9	7.13	22.8	17.40	17.8	6.93	10.00	17.9
10	11.00	12.0	7.89	21.5	15.90	13.50	18.3



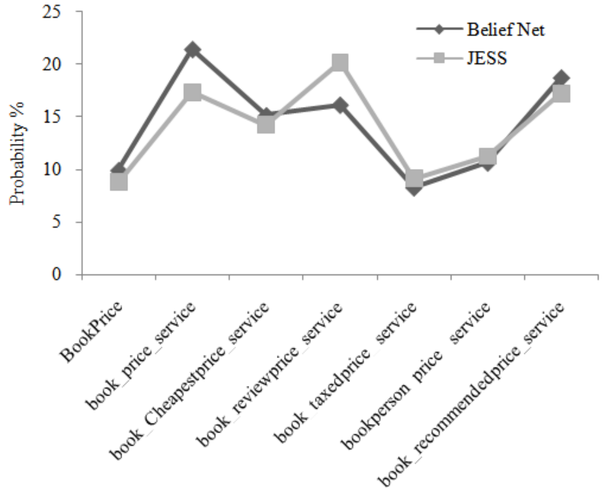
(a)



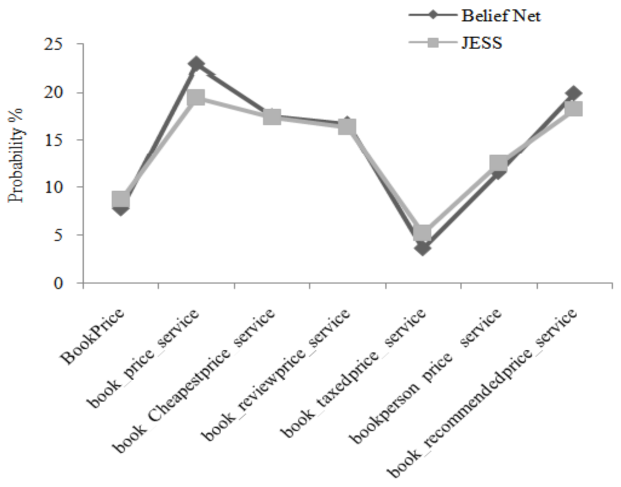
(b)



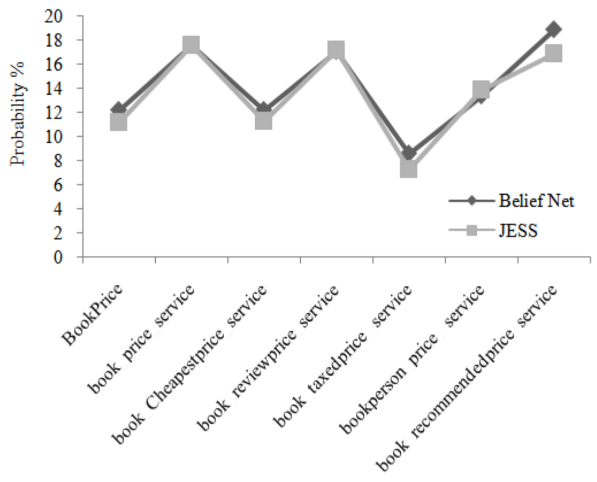
(c)



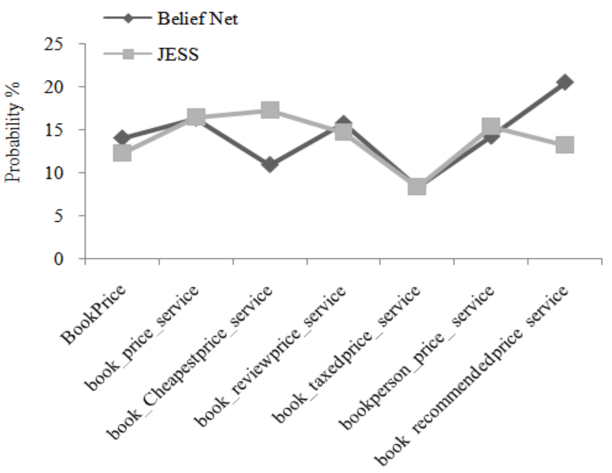
(d)



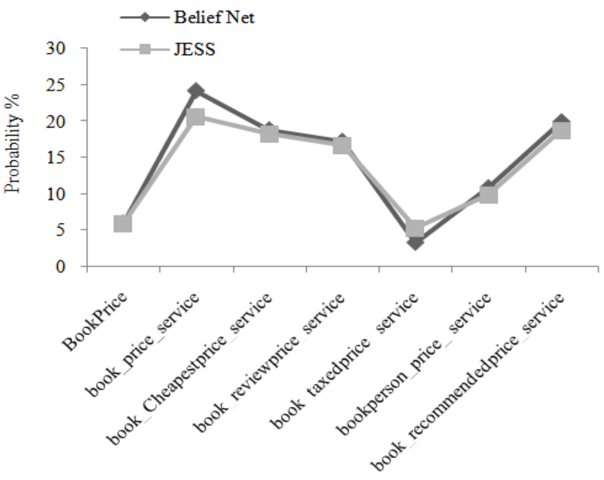
(e)



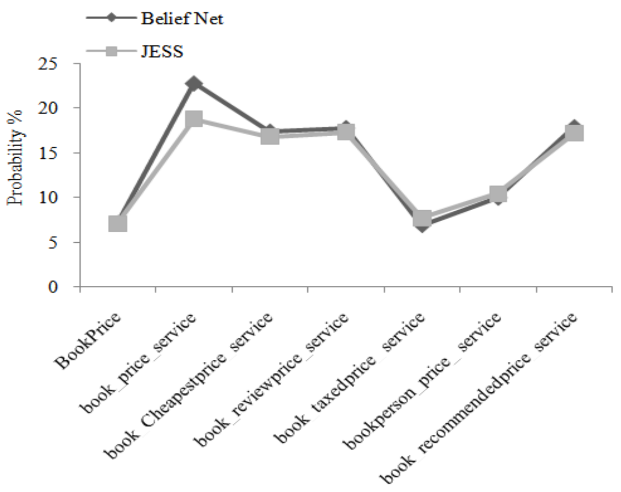
(f)



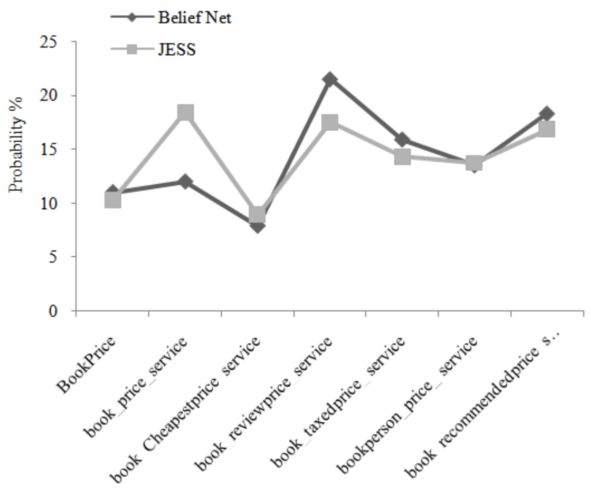
(g)



(h)



(i)



(j)

Fig. 5: The performance of belief network and JESS, a graphical representation of experimental results of (a) case 1, (b) case 2, (c) case 3, (c) case 4, (e) case 5, (f) case 6, (g) case 7, (h) case 8, (i) case 9, (j) case 10

CONCLUSION

In this study Web service discovery using JESS inference engine has been proposed. The experimental result shows that the JESS inference engine infers the potential web service which the service requestor wanted to use. The profile of the user is updated once with the user's preferences and can be used to make a fine tune search on the services of similar functionality. The profile is to be updated whenever the user wants to change the preference list; otherwise the same profile is used for several times and avoids each time entry into the system. The experimental results demonstrate the feasibility of our approach.

REFERENCES

Aklouf, Y. and E.K. Rezig, 2009. An ontological approach for dynamic functionality-based web services discovery using expert systems. Proceeding of the 2nd International Conference on the Applications of Digital Information and Web Technologies, (ICADIWT '09), pp: 187-192.

Al-Masri., E. and Q.H. Mahmoud, 2007. QoS-based discovery and ranking of web services. Proceedings of the 16th International Conference on Computer Communications and Networks (ICCCN'07), pp: 529-534.

David, M., B. Mark, H. Jerry, L. Ora, M. Drew, M. Sheila, N. Srini, P. Massimo, P. Bijan, P. Terry, S. Evren, S. Naveen and S. Katia, 2004. OWL-S: Semantic Markup for Web Services. Retrieved from: http://www.w3.1org/submission/owl-s/ (Accessed on: May 2, 2013).

Erik, C., C. Francisco, M. Greg and W. Sanjiva, 2001. WSDL Web Services Description Language. Retrieved from: http://www.w3.org/TR/2001/ NOTE-wsdl-20010315 (Accessed on: January 6, 2013).

Friedman-Hill, E.J., 2007. Jess-the Rule Engine for Java Platform. Retrieved from: http://herzberg.ca.sandia.gov/jess (Accessed on: May 6, 2014).

Gnanasekar, A. and R.M. Suresh, 2014. Content-based semantic web service discovery, an empirical analysis with implementation. J. Theor. Appl. Inform. Technol., 64(3): 635-640.

Gouscos, D., M. Kalikakis and P. Georgiadis, 2003. An approach to modeling Web service QoS and provision price. Proceeding of the 4th International Conference on Web Information Systems Engineering Workshops. Roma, Italy, pp: 121-130.

Gunasri, R. and R. Kanagaraj, 2014. Natural language processing and clustering based service discovery. Int. J. Sci. Technol. Res., 3(4): 28-31.

Harshavardhanan, P., J. Akilandeswari and R. Sarathkumar, 2012. Dynamic Web services discovery and selection using QoS-broker architecture. Proceeding of International Conference on Computer Communication and Informatics (ICCCI, 2012). Coimbatore, pp: 1-5.

Huang, A.F.M., C.W. Lan and S.J.H. Yang, 2009. An optimal QoS-based web service selection scheme. Inform. Sciences, 179(9): 3309-3322.

Kritikos, K. and D. Plexousakis, 2009. Requirements for Qos-based web service description and discovery. IEEE T. Serv. Comput., 2(4): 320-337.

Liangzhao, Z., B. Benatallah, A.H.H. Ngu, M. Dumas, J. Kalagnanam and H. Chang, 2004. QoS-aware middleware for web services composition. IEEE T. Software Eng., 30(5): 311-327.

Lin, S.Y., C.H. Lai, C.H. Wu and C.C. Lo, 2014. A trustworthy QoS-based collaborative filtering approach for web service discovery. J. Syst. Software, 93: 217-228.

Liu, M., M. Wang, W. Shen, N. Luo and J. Yan, 2012. A quality of service (QoS)-aware execution plan selection approach for a service composition process. Future Gener. Comp. Sy., 28(7): 1080-1089.

Liu, Z.Z., X. Xue, J.Q. Shen and W.R. Li, 2013. Web service dynamic composition based on decomposition of global QoS constraints. Int. J. Adv. Manuf. Tech., 69: 2247-2260.

Luc, C., H. Andrew, R. Claus and R. Tony, 2004. Universal Description Discovery and Integration (UDDI) Version 3.0.2. Retrieved from: http://www.uddi.org/ pubs/ uddi-v3.0.2-20041019.htm (Accessed on: May 5, 2013).

Maximilien, E.M. and M.P. Singh, 2004. A framework and ontology for dynamic Web services selection. IEEE Internet Comput., 8(5): 84-93.

Mukhopadhyay, D. and A. Chougule, 2012. A survey on web service discovery approaches. Adv. Comput. Sci. Eng. Appl., 166: 1001-1012.

Nair, M.K and V. Gopalakrishna, 2010. Look before you leap: A survey of web service discovery. Int. J. Comput. Appl., 7(5): 22-30.

Ran, S., 2003. A model for web services discovery with QoS. ACM SIGecom Exchanges, 4(1): 1-10.

Torres, R., H. Astudillo and R. Salas, 2011. Self-adaptive fuzzy QoS-driven web service discovery. Proceeding of IEEE International Conference on Services Computing (SCC), pp: 64-71.

Yang, S.J.H., J. Zhang and I.Y.L. Chen, 2008. A JESS-enabled context elicitation system for providing context-aware Web services. Expert Syst. Appl., 34(4): 2254-2266.