Agent Based Approach for Searching Document in Semantic Web

Wawan Laksito YS [1], Teguh Susyanto[2] and Andriani KKW [3]

[1] Informatics technique department, STMIK Sinar Nusantara Surakarta, Jawa Tengah, Indonesia

[2] Informatics technique department, STMIK Sinar Nusantara Surakarta, Jawa Tengah, Indonesia

[3] Information system department, STMIK Sinar Nusantara Surakarta, Jawa Tengah, Indonesia

Abstract

The use of web service technology has increased and spread rapidly in many places providing dynamic information to meet the business goal of each provider. However, all of the existing facilities nowadays are just for the direct users (human) and it is still impossible for its users (machine) to consume it. It is hoped that the lack of the service provided in the conventional web service can be overcomed by using the semantic web services. Therefore the writer proposes the making of an application framework to integrate the web services in searching information of documents in many platforms. The application framework, which is formed, functions as a provider of the document information service by using intelligent agent technology and the semantic web services on a distributed application.

Keywords: *Web service, Intelligent Agent, Semantic Web Services.*

1. Introduction

Nowadays, as the bigger and bigger increasing in various information service, it surely needs a technology which is able to connect and serve any information in relevant format and which is also suitable for its customer's need. The problem frequently met by the users in finding out the wanted information using Search engine is the process of filtering information produced by the service of internet application that has various and complex format, therefore to get the most suitable information it needs an extra time and struggle although sometimes it makes the users frustrated. This is due to the information provided in internet is specially designed for direct users between human and machine in which the process of the being browsed information selection is based on the matching of the searching key syntax and the result of the provided information still needs the help of the human [1].

Beside that, the use of web service technology has increased and been spread in many places to provide dynamic information in the form of distributed application to meet the business needs of each provider. To use the web service provider, the user must know the given service description available in the *WSDL (Web Service Description Language)* file. Meanwhile, to find the service location, a facility called *UDDI (Universal Discovery Descriptor Interface)* is provided. However, all of those facilities are still for the direct users (human) not to be consumed by the users (machine)[2]. It is necessary to provide a certain mechanism that can be understood by machine, that is service description (*web service)* formed in semantics. An agent application can use the service provided by provider by using semantic format (*ontology)* starting from the process of searching service, the service composition until *invocation* that result in the needed information as the output.

In this paper, the writer porposes an application of artificial agent software and semantic web service to solve the problem of searching document applicable in distributed application.

Based on this problem, this paper is aimed at creating an application framework to integrate web service to search document information in many platforms.

Therefore, the writer proposes the problems in this research as follows: (1) how to design a semantic model on web service that supports the process of the searching of information provider by using agent application, (2) how to make a system application of provider's document information service by using intelligent agent technology and *semantic web services* in a distributed application.

2. Related Works

Semantic web services is used to develop a traditional web service technology by consolidating ontology in semantics. Therefore, it is hoped that the integration and the calling of the sevice can be carried out dynamically. The semantic web service discovery that is developed by combining algorithm matching is for the process of searching services by testing the similarity level by involving the ontology domain [3] [4]. The OWL-S technology is used to support the semantic web service discovery to describe the web service in semantics. Through this semantic description, an artificial agent can 'understand' the concept defined in semantic web services, then extract the given back information and determine the service needed for the client [5].

Moreover, by using artificial agent that uses communication language called *Agent Communication Language (ACL)* with ontology developed from the standard *OWL* approach to create a *retrieval* data system so that it can be more flexible in identifying the agent's role in communication and negotiation [6].

3. Theory Background

3.1 Agent and Intelligent Agent

Agent is a computer system located in a certain environment and it can act autonomously in its environment to meet the given goal. Wooldridge differs between agent and intelligent agent that are necessary to be reactive, proactive, and social in advance [7].

An agent is autonomous because it operates without the human's direct role or other agents but by having control over internal execution and condition. An agent is social since it works together with human or other agents to meet its task. Agent is reactive since it can feel its environment and respond accurately to the changing occured in its environment. An agent is proactive because it does not only act in responding its environment but also can show the behavior that goes to the goal based on the taken inisiative. Beside that, an agent is also mobile, it has a capability to travel between different nodes in a computer network [8].

3.2 Semantic Web Services

The semantic web services technology is developed from the base of web services and semantic web. The web services offers an approach promising to reach the process of *loosely coupling* that crosses the organization limits. The web service technology serves specification that covers the detailed info needed to automatic interoperation between clients (agents) and the service in the web, by using a little role of human agent [9].

The service web is said to be a prominent technology in meeting a flexible solution especially to run an interoperation of several application system. The great number of the increasing in the available web service show a callenge on how the process of relevant *service discovery* becomes a question deserved to be answered. Although nowadays the process of the service searching has been *able* to be met through *public UDDI* but it is still not enough to be as the media of computer interpretation to support the web service finding [10].

Besides the ability of doing the service discovery, the semantic web services may also be able to do the *service composition, service invocation, and service monitoring* that can be done dynamically. Software agent has been thought as the potential user of the Semantic Web Services related with the interaction with *semantic description* of Semantic web services to support the process of discovery independently, the selection, the composing, the calling, and the execution which are based on the services to meet the user's need [11].

3.3 OWL-S

OWL-S is a specification to describe the semantic web services in ontology and is also the development of DAML-S specification developed as the part of DAML+OIL [12]. The three main parts of OWL-S are *Service model* that covers the description of the functional parameter definition, the format of the service interaction with the Invoker, and the mechanism of its execution in the composite things. Therefore, OWL-S serves (but not limited on) *the process model.* The *process model* explains the services as the process with the functional parameters: *Input, Output, Precondition, and Effect (IOPE).* These also determine the components of the process of the composite service and the command of execution and it bounds the input and the output of the component process.

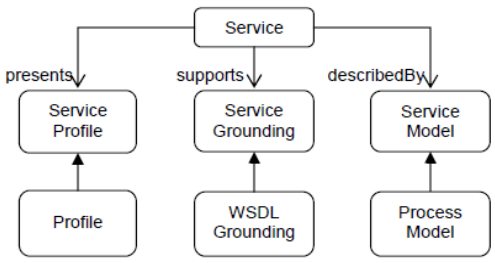

Figure 1. The main concept of OWL-S [9]

4. The Design of The Application

The architecture of the designed application is as shown on Picture 2. It consists of three layers of the main component.

a. *User interface layer*
User interface layer is functioned as an interface that bridges the user and the components of the other applications. The components in this layer serve an *input* media from a user to find the suitable service and information of the execution result from *web service* through an agent software courier in *middle layer*.

b. *Middle layer*
This layer functions as the business process roled by an agent software, containing *A Service Web Searching Agent* (Requester Agent) that is responsible for finding the service in the suitable service web, meanwhile A Web Service Caller Agent functions to carry out the service web calling process that is as the result of the *Web Service Searching Agent*. The finding of the service web will be carried out by matching process with algorithm *matchmaking* by Agent-Matchmaker. This Algorithm compares the inputed searching key specification by the user with the service web description represented by the semantics (*Ontology*). The result of the *Web Services* will be communicated to Agent-requester that will continue the calling process of the service web (*invoking)*. The communication between agent through FIPA-ACL (*Foundation for Intelligent Physical Agents - Agent Communication Language)* in the field of *JADE platform.*

c. *Knowledge Database Layer*
This layer is functioned as the base of application knowledge that can be used by software agent in carrying out the searching operation and the calling of *web service.* The knowledge base containing the information of the *web service* description in the form of semantic language describes the data scheme in the domain of document archive, especially to provide ontology information in the process of matching the class hierarchy (*subsumption)* with the algorithm of matchmaking between the inputed searching key with the description of the web service that is kept in *database registry.* The database registry saves the detail description of a group of the web service that has been registered.

4.1 The Agent Specification

The agents involved in the composed system contains Requester Agent and Matchmaker Agent whose roles are aimed at minimalizing the human's intervention.

4.1.1 Requester Agent

Requester Agent is an agent that is closely related with the user (human) and has a user interface to trigger the searching process. Here are the detailed information of the requester agent's roles:

- Providing interface to trigger the process of searching, that is by inputing the used Ontology URL domain, the detail information of the parameter type list that can be selected by users to determine the specification of the seb service going to be searched (*Capability Search)*
- Sending resquest specification that has been determined by the user to Matchmaker Agent
- Making the selection of the relevant web service possible by theuser and receiving the data paramater inputing (value) for the need of the web service calling process.
- Displaying the result of the service calling (*grounding*) to the user interface (GUI)

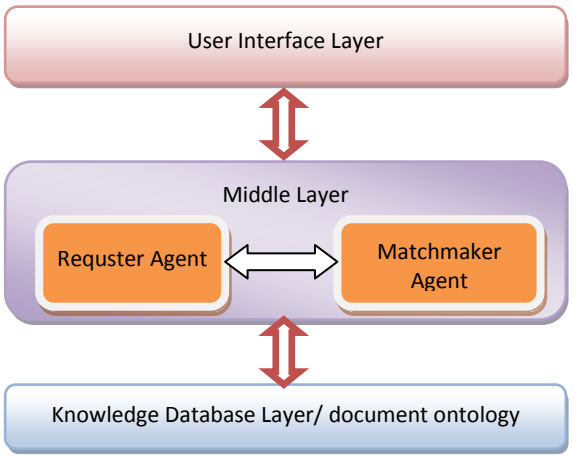

Figure 2. The architecture scheme of the application

4.1.2 Matchmaker Agent

The main function of a Matchmaker Agent is to do the searching of the most relevant service web suitable with the parameter specification inputed by the user (*query*). The parameter specification includes the being used ontology domain (*.owl), and the determining of the input and output parameter which will be searched for. The detail of the process controlled by A matchmaker agent is:

- Receving the requested message from the Requester Agent (in the form of the searching specification), and extracting that message to RDF format.
- The extracting result will be loaded onto ontology model.

- Doing the matchmaking process to find out the most relevant service web description of several groups of the service web registered in the storage media requested by the user.
- Returning the result of the most relevant service web back to the Requester Agent
- The levels of the sematic compatibility in matchmaking are:
  - **EXACT**, if the type of the query's input/output parameter is the same as the parameter of the candidate
  - **PLUGIN**, if the type of the query's input/output has higher hierarchy than the candidate's parameter
  - **SUBSUME**, if the candidate's input/output parameter has higher hierarchy than the query's parameter type
  - **FAIL**, if the type of the input/output parameter does not match with the candidate's parameter type.

The application going to be built has the flow of work as shown on figure 3. The application's flowchart is:

a. The provider through the OWL-S UI-Registered registers the service web to *database registry* in owl address format (*.owl). there has been a mapping of input and output description with the used ontology domain in this owl document.
b. The user inputs the keyword of searching through UI-Search Web Service in the form of detail specification of the fucntion parameter type that is going to be searched and sent to the Matchmaker-Agent.
c. The searching keyword will be translated into semantic format and processed by mathcmaker agent to find the most appropriate web service. In this case, computation of class hierarchy matching will take place based on the ontology domain in database knowledge by using matchmaking algorithm.
d. The description list of the web service in the form of owl document (*.owl) which is relevant to the searching result will be communicatd to the requester agent for the process of web service selection that will be executed by the user.
e. The reqeuester agent will receive the message sent by the matchmaker-agent in the form of seb service (*.owl) suitable with what is being asked for through ACL Message and display it to user's interface.
f. The user determines one of the web service descriptions (*.owl) that will be executed (invoked) by the requester-Agent. The calling process of the web service by the requester agent needs the input of the parameter's score data by the user. The result of the calling is in the form of document information and served in the user's interface.

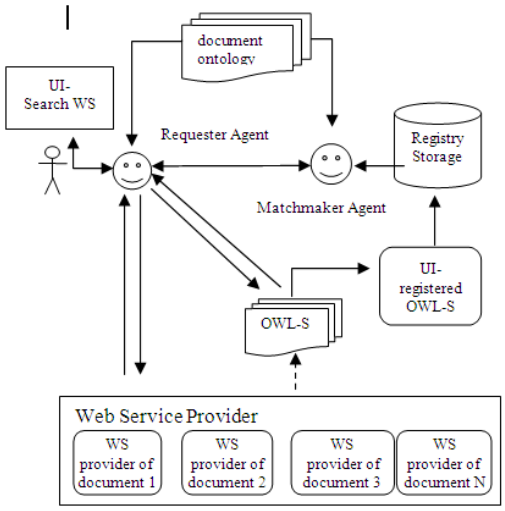

Figure 3. The application flowchart

5. Discussion

5.1 Ontology of Document

The document's ontology domain is used to model the document concept used as the semantice reference of terminologies going to be used by the web service. This model enables every being used terminologies can be determined its meaning/semantics, although it has several differences from its sintax side.

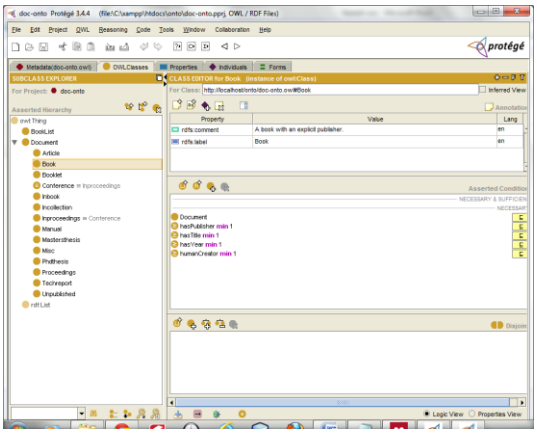

Figure 4. Document Ontology

The Domain ontology is made by using *Protege* application tool consisting of a group of Class concept (*Class)* and *property* (property data or property object). The formed class concepts include *Document* class and *Document's* derivative classes such as *Article*, *Book*, *Booklet*, *ConferenceBook*,

Inbook, *Incollection*, *Inproceedings*, *Manual*, *Masterthesis*, *Misc*, *Proceeding*, *Techreport*, *Unpublished*, and *PhdThesis*.

5.2 OWL-S Registration

To support the automation of the web service searching dynamically, every conducted web service must have semantic description in owl-s format because without it the being searched web service will not be found. Therefore every *owl-s* description must be collected in the media storage. This is aimed at making the query process of the web service searching efficient. To support it, the registration of owl-s interface is implemented (*SWS Registry Form).*

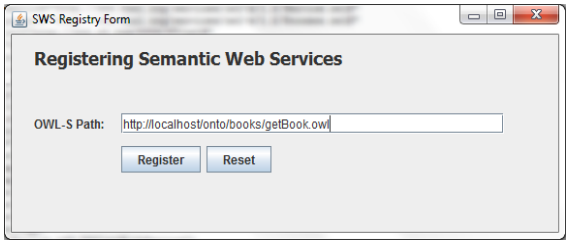

Figure 5. SWS Registry Form

The process of the owl-s description storage is defined as "wsregistry" database containing table of *mainregistry* and *servicedetail*.

5.3 The Searching Process of Web Service (*Discovery Service)*

For the effectivity of the document searching result by the user, there are two sstages conducted by the system:

- Stage 1 (*capability Search),* determining the document's specification which is going to be searched, in this case the user must determines the input type that is going to be inputed and the wanted output type. For example, the user wants to search book based on *isbn* as its searching attribute, then the user's request will be translated in query format for the web service searching that has *isbn* attribute type and the output type is Book, as what will be shown on figure 6.
- Stage 2 (Selection and Invoking Service)
  After the web service with the matched specification as requested has been found, then the application will display the relevant owl-s web service list. After that, the user inputs the input's attribute data such as ISBN = "979-3469-31-5" for the searching process of one os the chosen web services, and the execution result will display the information of the wanted Book searching (Book)

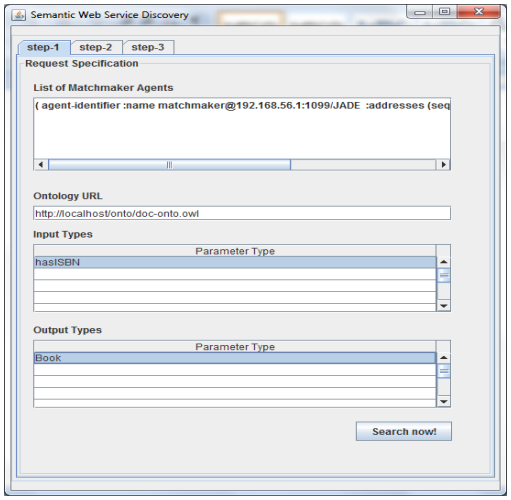

Figure 6. SWS Discovery Display

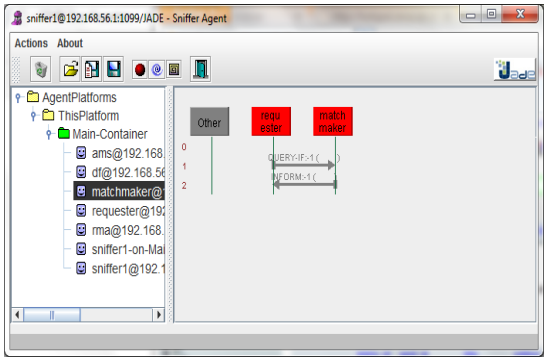

Figure 7. Sniffer Agent.

6. Testing

To get the best result suitable with the purpose of the research, here is the built application prototype testing. The testing which is is carried out refers to how effective and efficient the work of the built application prototype is.

6.1 The Testing of Matching Services

The testing of the Matching services is to measure the ontology matching process on the relevant web services by determining the specification of the input's parameter type and its semantic level.

```
Case 1 :
Request :
Parameter INPUT: hasISBN
Parameter OUTPUT: Book
Response:
Match: EXACT
ServiceName: getBook
Parameter Input: isbn:hasISBN
Parameter Output: book:Book

Case 2:
Request :
Parameter INPUT: hasISBN
```

```
Parameter OUTPUT: Document
Response:
Match: PLUGIN
ServiceName: getBook
Parameter Input:  isbn:hasISBN
Parameter Output: book:Book

Case 3:
Request :
Parameter INPUT: hasTitle
Parameter OUTPUT: Document
Response:
Match: FAIL
ServiceName: tidak ada
Parameter Input: tidak ada
Parameter Output: tidak ada

Case  4:
Request :
Parameter INPUT-1: hasAuthor
Parameter INPUT-2: hasYear
Parameter OUTPUT: BookList
Response:
Match: EXACT
ServiceName: searchBooksByAuthorAndYear
Parameter Input-1:  year:hasYear
Parameter Input-2:   author:hasAuthor
Parameter Output: books:BookList

Case 5:
Request :
Parameter INPUT-1: hasYear
Parameter INPUT-2: hasAuthor
Parameter OUTPUT: BookList
Response:
Match: EXACT
ServiceName: searchBooksByAuthorAndYear
Parameter Input 1:  year:hasYear
Parameter Input 2: author:hasAuthor
Parameter Output:  books:BookList
```

Based on the sample of *matching service* testing, the built application prototype has been able to show the result of the wished matching process. Testing of case 1, 4, and 5, show the matching level of EXACT that means the inputed request parameter has the same semantics as one of the web service ontologies in the registry database. Although the position of the request parameter (if more than one) is not matched with the parameter's position in the searched web service. In Case 2 testing, the system has been able to display the matching level of PLUGIN, This is due to the parameter type of the request input (query)is *Superclass* of the parameter type of the candidate service. Case 4 Testing shows FAIL, if the being serached ontology web service does not have similarity with all ontology stored in the registry database.

6.2 Performance Testing

For the need of quantitative analyze, a testing on the comparison between the number of time for processing the service web searching and the number of the collected service web ontology is carried out. The result of the testing is illustrated in table 1.

Tabel 1: The comparison between the web service searching time

Testing	Number of Service	Time (second)
1	6	3.264615192
2	8	3.289238033
3	10	3.188956105
4	12	3.240091421
5	14	3.308786779
6	16	3.134642471
7	18	3.603153051

There is no definite time difference between the testing of time based *matching service* and the number of service saved in the registry database. The cause is probably that every stage of testing does not have big difference on the number of the service in which every of them only has two difference service web.

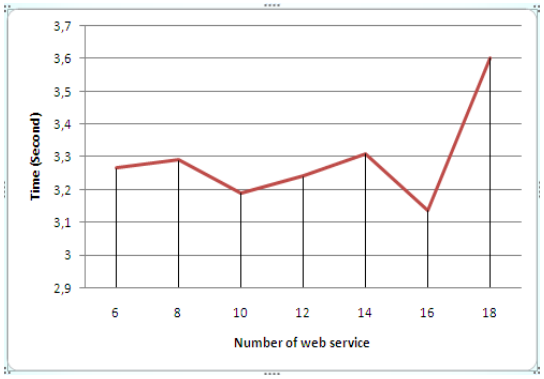

Figure 8. The graph of the web servince searching performance.

7. Conclusion

The use of semantic web service and multi-agents to support the process of the document searching results in several conclusions as follows:

- Various web services provide information service of document spread in various address and have different data formats that can be integrated by giving semantic anotation on the parameter type format of input and output in the web service described on the defined ontology domain.
- The process of searching information of a document is carried out on to matching stages; functional web service (*capability searching)* and data matching of the chosen web service execution result (*invoking service)*
- The defined document ontology must be able to accomodate every specification of registered web service.

- To support the performance of the autonomous system in the stages of the document searching process, requester agent and matchmaker agent are used. These agents perform as the courier (*broker*) to provide information service in the system.
- The process of matching this semantic web service has four matching levels, they are *Exact*, *Plugin*, *Subsume*, and *Fail*.

8. Future Work

To make the result of this research perfect, the writer recommends:

- Testing on web service performance must use more number of web service sample so that the resulted time difference is significant to measure the score of the scalability.
- It is very necessary to have deeper reseach especially on describing web service ontology with complex type.
- To optimilize the use of semantice web service technology, it is necessary to inlove the service composition dinamically.

References

[1] L. Yu, *Introduction to the Semantic Web and Semantic Web Services*. Chapman and Hall/CRC, 2007, p. 368.

[2] G. Wang, D. Xu, Y. Qi, and D. Hou, "A Semantic Match Algorithm for Web Services Based on Improved Semantic Distance," *2008 4th International Conference on Next Generation Web Services Practices*, pp. 101–106, Oct. 2008.

[3] L. Zhou, "An Approach of Semantic Web Service Discovery," *2010 International Conference on Communications and Mobile Computing*, pp. 537–540, Apr. 2010.

[4] Z.-H. Zeng and S. Ying, "The Usage of Semantic Condition Expression for Semantic Web Service Matchmaking," *2008 International Symposium on Intelligent Information Technology Application Workshops*, pp. 243–246, Dec. 2008.

[5] D. Çelik and A. Elci, "Searching Semantic Web Services: An Intelligent Agent Approach Using Semantic Enhancement of Client Input Term (s) and Matchmaking Step," *... Conference on Intelligent Agents, Web ...*, vol. 2, pp. 916–922, 2005.

[6] A. Diosteanu and L. Cotfas, "Agent Based Knowledge Management Solution using Ontology, Semantic Web Services and GIS," *Knowledge Management*, vol. 13, no. 4, pp. 90–98, 2009.

[7] M. Wooldridge, *An Introduction to Multiagent Systems 2nd Edition*. John Wiley & Sons, 2009.

[8] F. Bellifemine, G. Caire, and D. Greenwood, *Developing Multi-Agent Systems with JADE*. Wiley, 2007, p. 300.

[9] F. Hakimpour, S. Cong, and D. Damm, "A Practical Tutorial on Semantic Web Services," *hakimpour.com*.

[10] V. Kaewmarin, N. Arch-int, and S. Arch-int, "Semantic Web Service Discovery and Integration Using Service Search Crawler," *2008 International Conference on Computational Intelligence for Modelling Control & Automation*, pp. 884–888, 2008.

[11] M. Shafiq, Y. Ding, and D. Fensel, "Bridging multi agent systems and web services: towards interoperability between software agents and semantic web services," *Enterprise Distributed Object ...*, pp. 85–96, Oct. 2006.

[12] "OWL-S: Semantic Markup for Web Services," 2004. [Online]. Available: http://www.w3.org/Submission/OWL-S/.

Wawan Laksito YS is a lecturer of Informatics technique department in STMIK Sinar Nusantara, Surakarta, Indonesia. He graduated from Master Degree program in the field of computer at Dian Nuswantoro University, Semarang, Indonesia (UDINUS).

Teguh Susyanto is a lecturer of Informatics technique department in STMIK Sinar Nusantara Surakarta. He is still continuing his study in Master Degree program in the field of computer at Gadjah Mada University, Yogyakarta, Indonesia.

Andriani KKW is a lecturer of Information system department in STMIK Sinar Nusantara, Surakarta, Indonesia. She graduated from Master Degree program in the field of computer at Dian Nuswantoro University, Semarang, Indonesia (UDINUS).