DISCOVERY AND INTEGRATION OF JOB MARKET SERVICE USING SEMANTIC WEB SERVICE APPROACH

TEGUH SUSYANTO, ZUHRAH
1Department of Information System, STMIK Sinar Nusantara, Surakarta, INDONESIA
2Department of Information System, STMIK Sinar Nusantara, Surakarta, INDONESIA
E-mail: teguhsusyanto@gmail.com, zuhrah@sinus.ac.id

ABSTRACT

Nowadays, there have been many websites applying information services of job vacancy. However, they still have many weaknesses and have not been able to provide relevant information for the job seekers yet. One of the weaknesses is that there have not been any information sharing with other job fair websites, therefore the information of the job being offered is still very limited. The presence of semantic web service technology makes any web services possible to be integrated dynamically. It is now possible to do the process of discovery, composition, invocation, and monitoring of web service automatically. This paper is using the technology of semantic web service to integrate several services on job fair information in the computer network. The conducted research was focused on the service discovery to support the searching and selecting of the relevant and reliable web service of job market. The method applied on the service discovery is combining the algorithm of semantic matching and simple additive weighting. The result of the testing is that the relevant and the best web service is found out based on the calculation of the similarity of the functional parameter and the service quality.

Keywords: Web Service Integration, Semantic Web Service, Service Discovery, Job Market, Semantic Matching

1. INTRODUCTION

There have been many applied web services to spread information about job vacancy. The services have been attached on the websites of online job fair offering job information to the job seekers [1]. However, the present service of job market information still has many lacks of informing relevant job as wanted by the job seekers. It is because there is no service integration yet and not any information sharing with other job fair websites so that the recommended job vacancies do not provide other alternative of job selection.

Semantic web service is the development of semantic web technology and web service. It is developed to construct the description of web service in semantic language [2]. With semantic web service, various web services can be integrated dynamically so that it is now possible to run the process of discovery, composition, invocation, and monitoring of web service automatically.

A prototype of service discovery application based on functional parameter on web service and quality of service (QoS) is proposed on this paper using the combination of Semantic Matching and Simple Additive Weighting (SAW) algorithms. The goal of constructing this system is to gain relevant result of the searching of web service.

This paper proposed a prototype of service discovery application based on functional parameter of web service and quality of service (QoS) using combination of Semantic Matching and Simple Additive Weighting algorithms. The goal of constructing this system is to result in the relevant method of the searching of the web service.

The research was focused on the development of the searching and the execution of job market web service to support the dynamic integration model of the job market web service.

The result of this study is hoped to be beneficial for the job vacancy provider in making online presentations of job information easily. Beside that, the job seeker will also find information on job vacancies best suited his need easily.

2. LITERATURE REVIEW

Technology of Semantic Web Service have been used to integrate various service information on the
area of distributed networking. Paper [3] has proposed a broker application as the search engine by using semantic technology of web service. Paper [4], [5], and [6] proposed an integrated information service in health. The integration of information constructed in each paper have supported the service of discovery, the selection and execution, and the monitoring of service.

The similarity between Paper [3], [4], [5], and [6] and the proposed paper is on the use of the four level semantic matching algorithm. It is considered to be relevant enough in finding out the web service based on its functional parameter. However, it is not able to determine the web service based on the best order. The difference between this paper and the other researches is that the service discovery mechanism proposed in this paper was conducted in two processes by using Semantic Matching and Simple Additive Weighting. The first process was the determination of the relevant web service based on the calculation of the web service functional parameter. The second process was ordering the best to the least web service by using Simple Additive Weighting algorithm. The ranking of the web service was based on the aspect of the Quality of Service covering the compatibility of the functional parameter, the average accessing time and hit count.

3. THEORY BACKGROUND

3.1 Semantic Web Service

Semantic web services (SWS) is the development of the semantic web and web service technology which composes a mechanism understood by agent, that is by describing the service (web service), so that the agent application can use the services provided by provider [2].

Web service is stated to be the most dominant in giving flexible solution especially to run an interoperation of various application system. There are so many addition on the number of the available web service that a challenge on how a relevant service discovery process becomes a question to be worth to answer appears. Although nowadays, the process of service discovery can be achieved by public Universal Description, Discovery and Integration (UDDI). The UDDI structure is not stable enough as a media of computer interaction to support the process of web service discovery [7].

In addition to its capability to run a service discovery, it is also possible for semantic web service to run service composition, service invocation, and service monitoring dynamically. Software agent is predicted to be the potential user of Semantic Web Service in relation with interaction with semantic description of Semantic Web Services to support the discovery process independently, the selection, the construction, the calling and the execution which are based on the service on the user’s need [8].

3.2 OWL-S

OWL-S is a specification to describe Web Services with ontology and the development of DAML-S specification as the part of DAML program [9].

The structure of Service ontology consists of three types of knowledge of service as shown on Figure 1, each of them has the following characteristics [10]:

- **Service Profile Class** is a part of Service class describing the profile of a service in the form of functional specification which includes Input, Output, Precondition, and Effects (IOPE).

- **Service Model Class,** is a part of Service class describing a process model run by a service.

- **Service Grounding Class,** provides things needed to use a service in detail.

3.3 Algorithm of Semantic Matching

Service discovery needs an algorithm that can measure the compatibility value between query and the offered web service semantically, so that the most suitable web service with the criterion is found. The matching process is done by measuring the degree of match on the service’s functional parameter (service capability) including parameters
of input, output, precondition, and effect. Semantic Web Service has a service description containing functional parameters stated in Service Profile, Service Process, and Service Grounding. However, a single web service (atomic service) only contains functional description in the form of input and output parameters. The following is the basic algorithm of semantic matching with four levels of compatibility as proposed by [11]. The proposed algorithm will compare the inputted OWL-S Query with the OWL-S advertisement stored in the storage place (registry). The matching between query and advertisement, in which every compared input/output parameter must refer to the concepts defined in the context of ontology domain. The algorithm results in a group of service in which then is sorted based on every service’s degree of match.

```java
computeMatch(query){
    recordMatch = empty list
    forall adv in advertisements do {
        if match(query, adv) then
            recordMatch.append(query, adv)
    }
    return sort(recordMatch); }
```

**Figure 2: Main Control Loop [11]**

Main Control Loop of a matching service algorithm is illustrated on Figure 2, a query will be matched with all offered services (advertisement) stored in registry. If the query finds out the matching with several offered services, the service will be the result.

The matching between query and advertisement is based on the matching of all output query with all output advertisement and all input query with all input advertisement. The algorithm for the process of output matching is described on Figure 3. The success level of the matching is determined by the degree of match. If one of the output services does not match with the output advertisement, the matching process is considered fail.

```java
MATCHING(outQuery, outAdv) {
    globalDegreeMatch = EXACT
    forall outQ in outputsQuery do{
        if outA in outputsAdv such that
            degreeMatch = degreeOfMatch(outQ, outA)
            if (degreeMatch = fail) return FAIL
            if (degreeMatch < globalDegreeMatch) 
                globalDegreeMatch = degreeMatch
    }
    return globalDegreeMatch; }
```

**Figure 3: Semantic Matching Algorithm [11]**

The matching of the input parameters is also based on the same algorithm, however the order of the query and the advertisement is reversed. Meanwhile, the determination of degree of match uses algorithm as illustrated on Figure 4.

```java
degreeOfMatch(outQ, outA){
    if outA = outQ then return EXACT
    if outA subsume outQ then return PLUGIN
    if outQ subsume outA then return SUBSUME
    otherwise return FAIL;
}
```

**Figure 4: Degree of Match [11]**

The degree of match is divided into four levels of different value, namely: (i) EXACT, concept of outA has equivalent or the same type as outQ type based on ontology domain, (ii) PLUGIN, if outA subsume outQ, then it is assumed that if the concept of outQ type is super type of outA type based on the concept hierarchy on domain ontology. (iv) FAIL, if the condition of the two concepts is not fulfilled.

Four levels are ranked in chronological order involving: EXACT > PLUGIN > SUBSUME > FAIL. Based on this degree of match, the input or output of a web service is said to be relevant or not, only the FAIL value stating that the service does not match with query.

### 3.4 Algorithm of Simple Additive Weighting

Simple Additive Weighting (SAW) is often known as weighted addition. The basic concept of SAW method is searching for the weighted addition of the performance rating of every alternative on all attributes [12]. SAW model needs normalisation process of decision matrix to a scale that can be compared with all available alternative rating. The formula to do the normalisation uses Equation (1),

\[
    x_{ij} = \begin{cases} 
    \frac{x_{ij}}{\max_{i} x_{ij}} & \text{if } j \text{ is benefit} \\
    \frac{1}{\min_{i} x_{ij}} & \text{if } j \text{ is cost} 
    \end{cases}
\]  

(1)

In which \( r_{ij} \) is the rating of the performance normalized from alternative \( A_i \) on attribute \( C_j \), \( i=1,2,3,...,m \) and \( j=1,2,3,...,n \). The preference value of every alternative \( v_i \) is calculated by using Equation (2).

\[
    v_i = \sum_{j=1}^{n} w_j r_{ij}
\]  

(2)

The value of \( w_j \) is the weight given to attribute \( j \), while the value of \( v_n \), which is bigger, indicates that alternative \( A_i \) is selected.
4. RESEARCH METHODS

The paper’s independent variables are the profile of the web service and the quality of service. The description of web service profile contains functional information of the web service namely its input/output parameter, while the quality of service contains information of its reliability and quality including the average of accessing time and the accessing (hit) frequency.

Meanwhile, the dependent variable is the ability of the application system to provide relevant web service.

The research was conducted in five processes namely: (1) Analyzing the concepts in the job market domain schemed by ontology model. (2) constructing the web service description by converting the WSDL description to the OWL-S language format from the web service groups that will be registered/integrated to the system. (3) composing functional module to calculate the proximity value between query and every web service in the repository. The matching process in the module used semantic matching algorithm. In this process, semantic matching algorithm was molded in application business layer which bridged the user interface layer and the repository of the semantic web service. (4) Testing result of the research through application simulation based on the functional parameter of the web service. The testing was done by inputting a number of different queries into the targetted web service that had been registered to the repository. The performance testing was done by recording the processing time of web service matching in the repository.

5. THE PROPOSED SYSTEM

4.1 System Architecture

The approach of Semantic web service is used to integrate the web service of job market in internet with application which is constructed dynamically. The dynamic integration is the access bridge of the job seekers to any providers of job vacancy information. Technically, the dynamic integration is done by making additional description on each web service. The description is by giving semantic annotation to WSDL functional attribute like input and output parameters. The making of WSDL attribute annotation is related with the concepts defined in the ontology domain of the job market. The result of the semantic annotation is able to correlate every web service providing jobs although each of them has its own way to represent different information.

The designed architecture has four layers of application component as shown on Figure 5, containing: (a) Layer of User Interface, which is functioned as an interface to bridge the users and the components of the other application. It is used as the input of the searching process of the web service and to display the job information gained from the web service execution. (b) Layer of middleware, contains a number of module consisting of: (i) Registry module, used to manage the registration of the web service in the system, (ii) Query module, functioned to construct queries of the web service searching to make semantic matching possible to be done, (iii) Service module matching, functioned to calculate the value of proximity between the inputted query and the service stored in the database of service registry, (c) Layer of database provider which contains two database namely database of service registry and job ontology. Database of service registry is used to manage the profile of the web service including the name of the service, parameters of input/output and quality of service. Ontology of Job Market is the scheme constructed from the concepts (vocabulary) in the domain of job market represented in OWL language. In this job market ontology, there is a relation between concepts hierarchically and the relation consisting of the title of the job, the field of the business, location, education level and department, and job experience. (d) Layer of web service provider, containing several web services providing job information integrated in a system.

Figure 5: Proposed System Architecture
The **Service Discovery** is aimed at gaining the relevant web service of information source to get groups of job information. The process is raised by sending a **query** which has specification in the form of functional parameter of web service either the input or the output parameter in which each of them, the input or the output parameter in the query will be compared with every input/output parameter in the web service that has been registered in **service Registry** to get the similarity value. The determination process of the similarity value is calculated by using algorithm four level **service matching** in which the details of the similarity value consists of **exact**, **plugin**, **subsumes**, and **fail** to result in the most suitable web service is done by using **Simple Additive Weighting** *(SAW)* based on attributes: the similarity value of input/output parameter, the average of execution time and **hit service**. The result of the calculation of all web services with the weighted addition is then ranked based on the gaining of the total **score** of every web services.

Ontology of the job market has been designed based on the identification result of the field fact gathering in the job market containing the design of **Class**, **ObjectProperty** and **DatatypeProperty**. The design result is shown in detail on Figure 6 covering **Class Industry**, **EducationDegree**, **EducationField**, **Location**, **JobOffer**, **Education**, **WorkExperience** and **ContactPerson** reusing from the class of foaf:Person.

To relate the relation between classes **ObjectProperty** stated in either the **domain** or the **range** is defined as shown on Figure 6. In detail, the relation between **ObjectProperty** and Class that have been designed covers: **hasJobField**, **hasLocation**, **hasCompany**, **hasPosition**, **hasGenderRequirement**, **hasEducationRequirement**, **hasDesiredJob**, **hasWorkExperienceRequirement**, **hasEducationHistory**, **hasWorkExperience**, **hasEducationDegree**, **hasEducationField**, **hasWorkExperienceField**, **hasWorkExperienceOccupation**, **hasDesiredJobField**, **hasDesiredJobLocation**, **hasDesiredJobPosition**.

Beside the concepts of **ObjectProperty**, **DatatypeProperty** as well as its domain and **range** has been defined to relate Class and Literal as shown on Figure 6, covering **hasIndustryName**, **hasDegreeName**, **hasEducationFieldName**, **hasLocationName**, **hasJobTitle**, **hasAgeRequirement**, **hasExpiredDate**, **hasGradePoint-Average**, **hasLongExperience** and so on.

### 4.2 Service discovery

Getting relevant web service is determined by the value gained from the weighted addition from **similarity parameter**, the value of **hit count**, and the average time of execution time by using SAW algorithm. Semantic matching algorithm is used to calculate the **similarity parameter**. It is used to gain the similarity value between query and the offered service. The similarity value is gained from the counting result of the total value of the similarity of the web service’s functional parameter between query and service. Attribute of **hit count** is the value of counter execution and the average attribute is the average time needed to execute a web service.

The similarity value of **service matching** is classified into four levels namely **EXACT, PLUGIN, SUBSUME, and FAIL**. the score of every level is then determined in chronological order 1, 0.75, 0.5 and 0. If the type of parameter **query** is equivalent with the type of service parameter, the score is **EXACT**, meanwhile if the type of **query parameter** is **subType** of the type of service parameter, the score is **SUBSUME**, and if the type of the **query parameter** is **superType** of the type of the service parameter, the score is **PLUGIN**. If the three conditions are not fulfilled, the score then is **FAIL**.

The flow of success on discovery service to find the **service**, first they are given a query as the input. **Query** contains the detail input and output parameters of the web service being looked for. Every web service’s parameter contains name and the type of the main parameter, and every type of parameters refers to the concept/vocabulary that has been formulated in ontology of the job market. Therefore the similarity value between the concepts (vocabulary) in the ontology will be gained during the matching process. Iteration process then is done by a number of **n** services stored in the **service registry**, and in every stage of the **n** iteration, the calculation of the similarity between the types of input/output parameters of the **advertisement service** and the type of input/output parameters of the **query is done**. Therefore, it results in a group of parameters which have similarity with query parameter. The next stage is determining whether the number of the output parameter fulfilling the **query parameter** is bigger than the number of the **n** service of parameter (**advOutput**), if it is true then iteration is continued to the next service without comparing input parameters. Otherwise, the comparison on the type of input parameters between **service** and **query** will be done. Based on
The comparison between every input parameter from the \( n \) service with query parameter, if the number of similarity on the parameter of the advertisement service input is bigger than the queryInput parameter, the iteration will be continued to the next service. On the other hand, if the parameter of the advertisement Input is smaller than or equals with the number of the query parameter, the being read \( n \) service will be inputted as the fulfilled service. All the service result fulfilling the query is grouped in the list of services.

On Table 1, eight samples of service registered to the service registry are presented. Each service is distinguished by input and output parameter attributes, the average execution time, and the hit services.

<table>
<thead>
<tr>
<th>ID</th>
<th>Service ( S )</th>
<th>Input (name: type)</th>
<th>Output (name: type)</th>
<th>Exec Time</th>
<th>Hit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>getJobs1</td>
<td>title: hasTitle</td>
<td>lsJobs: JobOffer</td>
<td>0.290</td>
<td>54</td>
</tr>
<tr>
<td>2</td>
<td>getJobs2</td>
<td>loc: hasLocationName</td>
<td>lsJobs: JobOffer</td>
<td>0.381</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>getJobs3</td>
<td>ind: hasIndustryName</td>
<td>lsJobs: JobOffer</td>
<td>0.716</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>jobByInd</td>
<td>Industry: hasIndustryName</td>
<td>JobOffer</td>
<td>0.351</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>jobByTtl</td>
<td>Title: hasJobTitle</td>
<td>JobOffer</td>
<td>0.282</td>
<td>153</td>
</tr>
<tr>
<td>6</td>
<td>jobByOcc</td>
<td>Occupation: hasOccupationName</td>
<td>JobOffer</td>
<td>0.421</td>
<td>12</td>
</tr>
<tr>
<td>7</td>
<td>jobByLoc</td>
<td>Location: hasLocatioName</td>
<td>JobOffer</td>
<td>0.311</td>
<td>14</td>
</tr>
<tr>
<td>8</td>
<td>jobByOcc AndInd</td>
<td>Occupation: hasOccupationName, Industry: hasIndustryName</td>
<td>JobOffer</td>
<td>0.452</td>
<td>11</td>
</tr>
</tbody>
</table>

Table 1: Advertisement Services

A query \( Q \) is given by a user to get the most suitable service \( S \) of which are illustrated in Table 1 with the input/output parameter specification as follows:

**Query Service: \( Q \)**

**Parameter Input**
- input-1: hasOccupationName
- input-2: hasIndustryName

**Parameter Output**
- output-1: JobOffer

The weight of the attribute is determined to search for service covering parameter = 0.6 hit = 0.2 and the average of execution time = 0.2. therefore, the matching process of all input/output parameters of service \( S \) with query \( Q \) will result in the similarity value as illustrated below:

1. **getJobs1**
   **Matching Output**
   PLUG IN: output-1: JobOffer - V.S- lsJobs: JobOffers
   **Matching Input**
   FAIL: title: hasJobTitle - V.S- input-1: hasIndustryName
   FAIL: title: hasJobTitle - V.S- input-2: hasOccupationName

2. **getJobs2**
   **Matching Output**
   PLUG IN: output-1: JobOffer - V.S- lsJobs: JobOffers
   **Matching Input**
   FAIL: loc: hasLocationName - V.S- input-1: hasIndustryName
   FAIL: loc: hasLocationName - V.S- input-2: hasOccupationName

3. **getJobs3**
   **Matching Output**
   PLUG IN: output-1: JobOffer - V.S- lsJobs: JobOffers
   **Matching Input**
   EXACT: ind: hasIndustryName - V.S- input-1: hasIndustryName
   FAIL: ind: hasIndustryName - V.S- input-2: hasOccupationName

4. **jobByInd**
   **Matching Output**
   PLUG IN: output-1: JobOffer - V.S- jobs: JobOffers
   **Matching Input**
   EXACT: industry: hasIndustryName - V.S- input-1: hasIndustryName
   FAIL: industry: hasIndustryName - V.S- input-2: hasOccupationName

5. **jobByTtl**
   **Matching Output**
   PLUG IN: output-1: JobOffer - V.S- jobs: JobOffers
   **Matching Input**
   EXACT: title: hasJobTitle - V.S- input-1: hasIndustryName
   FAIL: title: hasJobTitle - V.S- input-2: hasOccupationName

6. **jobByOcc**
   **Matching Output**
   PLUG IN: output-1: JobOffer - V.S- jobs: JobOffers
   **Matching Input**
   EXACT: occupation: hasOccupationName - V.S- input-1: hasIndustryName
   FAIL: occupation: hasOccupationName - V.S- input-2: hasOccupationName

7. **jobByLoc**
   **Matching Output**
   PLUG IN: output-1: JobOffer - V.S- jobs: JobOffers
   **Matching Input**
Based on the result of the matching parameter above, several services with the number of output $S_{out}$ parameter, except the value of FAIL is less than or the same as (≥) parameter output query $Q_{out}$ parameter are then selected and the number of input service $S_{in}$ parameter, except the value of FAIL is less than or the same as (≤) input query $Q_{in}$ parameter will be classified as the fulfilled service. In chronological order, the value of EXACT, PLUG_IN, SUBSUME, FAIL has point of 1, 0.75, 0.5 and 0. Therefore, the total value of parameter similarity on the service can be calculated:

**service: jobByOccAndInd**

$\text{param similarity} = \frac{\text{Average Input Parameter} + \text{Average Output Parameter}}{2}$

$\text{param similarity} = \frac{(0.75/1) + (1+1)/2}{2} = 0.875$

The determination result of the service $S$ with the most suitable parameter with query $Q$ is shown in Table 2.

<table>
<thead>
<tr>
<th>ID</th>
<th>Service (S)</th>
<th>Param Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>getJobs3</td>
<td>0.875</td>
</tr>
<tr>
<td>4</td>
<td>jobByInd</td>
<td>0.875</td>
</tr>
<tr>
<td>6</td>
<td>jobByOcc</td>
<td>0.875</td>
</tr>
<tr>
<td>8</td>
<td>jobByOccAndInd</td>
<td>0.875</td>
</tr>
</tbody>
</table>

The calculation result of the service parameter is processed by using algorithm of **Simple Additive Weighting (SAW)** based on the number of the parameter attribute, hit and the average time of execution. The four services on Table 2 are formed into $M$ matrix with the number of lines as many as four web services and three column each are the parameter of score, hit, and average time of execution in chronological order. $M$ matrices is then normalized by using Equation (1) for each attribute, so that an $r$ normal matrix is formed.

$$r = \begin{bmatrix}
0.0 & 0.83 & 0.49 \\
1.0 & 0.83 & 1.0 \\
1.0 & 1.0 & 0.83 \\
1.0 & 0.91 & 0.78
\end{bmatrix}$$

It is assumed, in chronological order, that the weight of the attribute parameter similarity, hit count and average time of execution is 0.6, 0.2, and 0.2 to construct $w$ matrix.

$$w = \begin{bmatrix}
0.6 & 0.2 & 0.2
\end{bmatrix}$$

Matrix $r$ and matrix weight $w$ are calculated by using Equation (2) so that the total score of each web service will be gained.

$$v = \begin{bmatrix}
1.0 & 0.6 + 0.83 \\
1.0 & 0.6 + 0.83 \\
1.0 & 0.6 + 0.91
\end{bmatrix} = \begin{bmatrix}
0.864 & 0.864 \\
0.966 & 0.966 \\
0.938 & 0.938
\end{bmatrix}$$

Hence, the result of the similarity calculation on service matching of the four services shows that the service with the highest degree of similarity is web services **jobByInd** and **jobByOcc** with their own score of 0.966.

6. DISCUSSION

The testing of service discovery is carried out by simulating application through giving searching query with variative functional parameter. On the five stages of testing, parameter of web service output valued PLUG_IN is gained which means the web service output parameter is the sub type (subsume) of the type of output parameter on the query. The similarity value of the web service input parameter is EXACT which means the type of input is identical (equal) with the parameter type of the input on the query. From the result of every testing, the service discovery is said to be valid in which there is no parameter with FAIL value in the displayed web service.

Most of the resulted web services have the same value of parameter similarities between one web service and the others. As an example, the first test case, there are two web services in which their input value of parameter similarities is EXACT (the value level = 1) and the value of output parameter similarities of PLUG_IN (degree level = 0.75), if the value of similarities of all output parameter is calculated, every web service resulted from the first test case has the value of $(1 + 0.75)/2 = 0.875$. Therefore, the next calculation is needed to make every displayed web service can be different from...
based on the best order. In this paper, SAW method was used to determine the best WS based on the value of parameter's similarity, the WS average time of execution and the hit value of calling for the web service. Therefore, the average time of execution and hit of calling for the web service is always renewed, so that WS which has the smallest average time and the biggest hit value has bigger opportunities to be the best web service. The time needed for every process of the service discovery is as many as 10 web services in which it is as much as 3.405 seconds.

The semantic matching process has run as it was hoped previously. It is proven by the result of the system testing which has been able to display the relevant web service although the keyword of the query searching uses different syntax mentioned in their own web service description. However, it has semantic relationship based on the defined ontology. The capability is the one that is not owned by UDDI in which the searching process will find out the result if the searching query has the same syntax as the web service description [13]. Beside that, the degree of match in UDDI’s discovery service only uses two levels, namely EXACT and FAIL. Meanwhile the discovery service in the research uses four degrees of match, namely EXACT, PLUG_IN, SUBSUME and FAIL.

7. CONCLUSION

Based on the result of the testing analysis, it can be concluded that the service discovery on the proposed system has been able to display all web services with similarity value of the functional parameter on the query. Beside that, the best web service is determined based on the biggest similarity value of the functional parameter, the biggest hit value and the smallest execution time. Meanwhile the time needed for counting the similarity value of the web service is around 0.34 seconds.

There are still many lacks in this paper, so that it is necessary to develop it to increase its performance, especially to add the ability of the Service Composition so that it is possible to handle complex query.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the contributions of The Ministry of Research, Technology and High Education, Republic of Indonesia, for funding this research.

REFERENCES:


Figure 6: Job Market Ontology