KURSOR Journal
Research on Computing and its Aplications

Vol. 7, No. 1, January 2013 **ISSN 0216 – 0544**

SEMANTIC WEB SERVICE COMPOSITIONFOR ERP BUSINESS PROCESS

[a]Anang Kunaefi, [b]Riyanarto Sarno, [c]Dwi Sunaryono, [d]Imam Mukhlash
[a,b,c,d] Institut Teknologi Sepuluh Nopember Surabaya
E-Mail: an_kunaefi@yahoo.co.id

Abstrak

Saat ini, ERP (Enterprise Resource Planning) bergerak menuju layanan SaaS (Software as a Service) dan Multi-Tenancy, di mana aplikasi ERP melayani beberapa penyewa dengan proses bisnis yang berbeda dalam lingkungan berbasisweb service. Pada kasus Provider ERP, sangat penting untuk mencapai fleksibilitas proses bisnis penyewa sebagaimana didefinisikan dalam tingkatkematangan SaaS level 4, yaitu Configurable dan Scalable. Dengan cara ini, Penyedia dapat melayani proses bisnis penyewa secara dinamis.Penelitian ini menggunakan pendekatan komposisi semantik web service untuk menyelesaikan masalah fleksibilitas dalamproses bisnis. Ontologi digunakan sebagai representasi pengetahuan semantik pada domainpengetahuan ERP untuk proses pencarian dan komposisi web service. Selanjutnya, algoritma kemiripan berbasis fitur (Feature-based Similarity) dan kemiripan berbasis struktur (Structure-based Similarity)digunakan untuk melakukan pencarian kemiripan antara permintaan proses bisnis dariPenyewa dan proses bisnis Penyedia layanan ERP di Registry. Hasil penelitian menunjukkan bahwa metode yang diusulkan mampu memenuhi permintaan proses bisnis penyewa, baik workflow sederhana maupun workflow yang lebih kompleks dengan hasil yang baik.

Kata kunci*:* Web Servis Semantik, Komposisi Semantik, Proses Bisnis ERP, *Feature-based Similarity, Structure-based Similarity.*

Abstract

Nowadays, ERP (Enterprise Resource Planning) moves toward SaaS (Software as a Service) and Multi-Tenancy, where an ERP application serves multiple tenants with different business processes in a web-service based environment. In the case of ERP provider, it is very important to achieve business process flexibility among tenants as defined in SaaS Maturity Level 4, that is Configurable and Scalable. InThis way, Provider can serve tenant's business processes request dynamically.This research usingsemantic Web Service Composition approach to address business process flexibility problem. Ontology is used as a semantic representation of ERP domain knowledge for web service discovery and composition. Afterwards, the combination of Feature-based Similarity and Structural-based Similarity algorithms are used to do the discovery and matchmaking process between tenant's business process request and business process available in the ERP provider's registry. The result showsthat the proposed method in this paper is able to fulfil tenant's business process request both for simple workflow and complex workflow with a good result.

Keywords: Semantic Web Service, Semantic Composition, ERP Business Process, Feature-based Similarity, Structure-based Similarity.

INTRODUCTION

Nowadays, ERP moves toward SaaS (Software as a Service) and Multi-Tenancy, where an ERP application serves multiple tenants with different business processes.In order to achieve flexibility, SaaSERP Provider needs a method to compose or configure business process at runtime.This requirement was defined by Qihongas SaaS Maturity Level 4 [1].

In this study, we use semantic web service composition to address business process flexibility problem. W3C defined Web Service Composition as a sequence of atomic services where one web service invoke another web service in order to make a new functional composite web service for its user.

In the previous studies about web service composition, some researchers used different methods and approaches to do the composition. Here are to name a few, Sirin used artificial intelligence approach to make service composition [2]. McIlraith used logical programming to automatically compose web services [3], and Talantikite used ontology annotation to compose web service to meet user's query [4]. Although many researchers have been addressed service composition problem with different techniques, composing a service from existing service is still very difficult [5]. This indicates that the world of service composition still in search for the best solution up until now.

Many studies used ontology to represent the metadata of the service, as ontology has been standardized by W3C [6,7,8]. However, for ontology approach alone, there are different methods used by researchers to do service composition. These methods are an extension of ontology specification designed for web service, for example, DAML as in [9,10], OWL-S as in [11,12,13], and WSMO as in [14,15].

Most researchers checked only the inputs and outputs of web services to make service composition [4,16,17,18]. By using this approach, the result of the composition can not be controlled by the user. Composition result can be very lengthy and inefficient, involving web services which should not be included.

In our study, we check the inputs and outputs (feature-based similarity), as well as the workflow structure (structure-based similarity) of the tenant's request. We believe, combining these two similarity checking will lead to a better composition result.

In a usual scenario, user or tenant of ERP Provider will perform a query to discover whether their service request is available or not in ERP Provider's registry before transaction take place.

In our scenario, however, tenant will perform a business process request using ontology annotation. Then,ERP Provider will perform similarity computation to searchfor similar composite service in the Registry. If, for example, the tenant's request cannot be satisfied with existing composite service, ERP Provider will try to compose a new composite service based on a composite service that has the closest match with tenant's request.

In order to accomplish this task, we explain our method in the next section.

DOMAIN ONTOLOGY

In heterogeneous environment, it is very important to use domain ontology as a reference for ontology processing.
Domain ontology will help to solve heterogeneity problem in the case of concept misperception [19]. For example, tenant might use term "Order" to express purchase order process. On the other hand, in Provider's side, terms "Order" might have at least two meanings, either "Purchase Order" or "Production Order". Purchase Order and Production Order are different in the perspective of Provider's Ontology, because they have their own characteristics. This condition is described in Figure. 1.

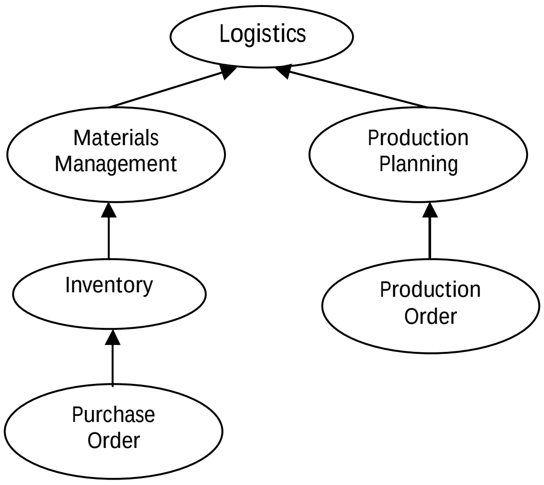
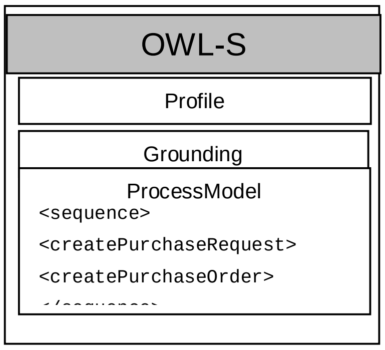


Figure 1. Example of ERP Domain Ontology.



```
<sequence>
<createPurchaseRequest>
<createPurchaseOrder>
```

Figure 2. ProcessModel Representation in OWL-S.

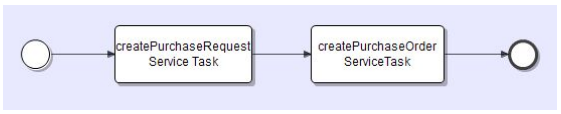

createPurchaseRequest Service Task
createPurchaseOrder ServiceTask

Figure 3. Graph Structure Representation

Using these characteristics, the system is able to determine which class in the ontology that has similar meaning with user or tenant's request, whether it is "Purchase Order" or "Production Order".

We adopted layered ontology approach explain in [20], by separating ontology for ERP domain knowledge (Domain Ontology) and ontology for ERP application (Application Ontology). The benefit of this separation is to achieve comprehensive understanding about the meaning, purpose and usage of each concept in the ontology [20].

The ontology was made by following practical guide in [21] using OWL 1.2 specification with Protege 4.2.

SEMANTIC SIMILARITY

In this research, we employ two kinds of web service semantic similarity algorithms, which are, Feature-based Similarity and Structure-based Similarity.

Feature-based Similarity

In the case of web service, Feature is defined as Input, Output, Precondition and Effect (IOPE). Although the technique might different, many researchers used IOPE matchmaking as a basis to perform service discovery and service composition [16,17].

Therefore, in this study, we adopt Feature-based Similarity to semantically discover whether two services can be composed or not. In Feature-based Similarity, the more features in common between the two concepts, the greater the similarity value. To calculate the Feature-based Similarity, we used the formula from Ganjisaffar [22] as shown in Equation (1),

$$\sigma(c,c') = \frac{2|F(c) \cap F(c')|}{|F(c) \cap F(c')| + |F(c) \cup F(c')|} \qquad (1)$$

Where,

F (c)	: Features of Concept *c*
F (c')	: Features of Concept *c'*

Structure-Based Similarity

In the case of ERP, each tenant can have many different business processes. Each business process is implemented using a composite service contain a process workflow. In OWL-S, this process workflow called as Process Model [23]. In OWL-S, the Process Model is represented in XML as described in Figure. 2.

In Figure.2, Process Model describes sequence of two web services i.e., *createPurchaseRequest* and *createPurchaseOrder*. This sequence can also be represented as a graph structure as described in Figure. 3.

The purpose of using Structure-based similarity is to calculate the similarity of structure and sequence of the workflow process [24,25]. Graph-edit Distance Algorithm is used to compute the distance between the two graphs. Graph-edit Distance is defined as the minimum number of graph-edit operations necessary for the two graphs to be exactly the same. The definition of graph-edit operations include: node deletion and insertion; node substitution; edge deletion and insertion.

The detail process about how to compute Graph-edit Distance Algorithm has been discussed in [24].

WEB SERVICE COMPOSITION

In a service composition scenario using Input, Output, Precondition and Effect (IOPE), two services S_1 and S_2 can be composed, if the output of S_1 is used as an input of S_2 [26]. Furthermore, two services are compatible and composable, if the `Effect` value of S_1 is equal or similar to `Precondition` value of S_2.

In addition, the structure of the composition request is matched with the structure of composite service available in the Registry. The objective is, a new composition can be made from existing composite service with minor addition.

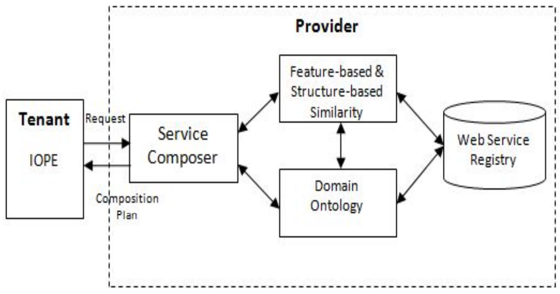


Figure 4. System Architecture.

In our method as described in Figure 4, tenant's business process request is handled by Service Composer. Service Composer utilizes Feature-based Similarity that is combined with Structure-based Similarity for matchmaking and composing workflow.For every similarity checking we use Domain Ontology to check the relatedness of every term of the service feature involved. Base on the result of matchmaking process, Service Composer return several Composition Plan that might be found to the Tenant.

Composing a Simple Workflow

Say, tenant define a request R as follows.

```
R={(input,purchaseOrderID,http://erp201
1/domain.owl#purchaseOrderID),
(output,stockStatus,http://erp2011/doma
in.owl#stockStatus),
(precondition,itemOrdered,http://erp201
1/domain.owl#itemOrdered),
(effect,purchaseInvoiceCreated,
http://erp2011/domain.owl#purchaseInvoi
ced),
(process,receiving,http://erp2011/recei
ving.owl}
```

Extracting Feature

Common web service features consist of Input, Output, Precondition and Effect (IOPE). For every feature in the above definition, there are two values separated with comma. The first value represent the name of the feature, the second value represent the annotation of the feature referencing to some ontology.

For example, `input` feature has value `purchaseOrderID` and `http://erp2011/domain.owl#purchaseOrderID`. The first value is the name of the input, and the second is the annotation ontology. In other words, tenant is searching for web service, where the input name is similar to `purchaseOrderID` with characteristics similar to what is described in `http://erp2011/domain.owl#purchaseOrderID`.

From the domain ontology specification `domain.owl#purchaseOrderID`, we get the following description about input `purchaseOrderID`.

```
F(purchaseOrderID) =
   {(hasInputName,purchaseOrderID),
   (hasInputType,String)}
```

Using the same steps as `input` feature, we get all features of request *R* as follows.

```
F(R) = {(hasInputName,purchaseOrderID),
       (hasInputType,String),
       (hasOutputName,stockStatus),
       (hasOutputType,String),
       (hasPrecondition,itemOrdered),
       (hasEffect,purchaseInvoiceCreated
       )}
```

After getting the features, the next important task is getting the structure of the request. Again, we will use ontology description that is annotated by tenant in the request. The system read and extract Process Model of composite service in ontology file `http://erp2011/receiving.owl`. Figure 5 described an example of receiving ontology as a part of Application Ontology.

In the ontology, there are four classes, and five object properties. Receiving class has three processes, namely, `createReceiving`, `checkQuantity` and `createPurchaseInvoice`. `Receiving` class has `hasInitialProcess` property on `createReceiving` class, indicating that `createReceiving` class is the first process that should be executed in the workflow. On the other hand, it also has `hasLastProcess` property pointing on `createPurchaseInvoice`, indicating that `createPurchaseInvoice` is the last process of the workflow. It also relates with `checkQuantity` using `hasProcess` property.

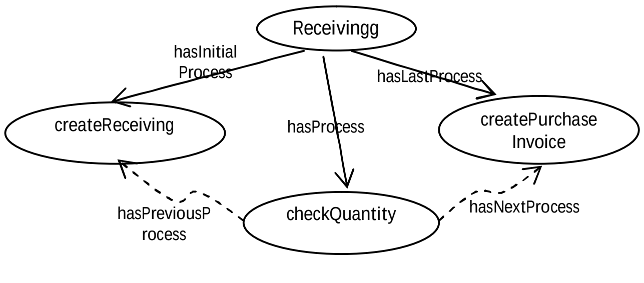


Figure 5. Example of Receiving.owl.

The position of `checkQuantity` process in the workflow can be determined with two properties, namely `hasPreviousProcess` and `hasNextProcess`. `hasPreviousProcess` means that `checkQuantity` can be executed after `createReceiving` is executed. On the contrary, `hasNextProcess` means that after executing `checkQuantity`, the system must execute `createPurchaseInvoice`.

In the bigger workflow involving more processes, there might be more than one `hasProcess` property relating with several classes. Each class should define its `hasPreviousProcess` and `hasNextProcess` property clearly. The structure graph as a result of extraction process of ontology in Figure 5 is shown in Figure 6.

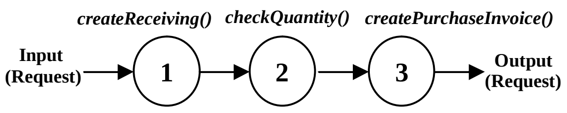


Figure 6. Graph Structure Representation of Composition Request.

Service Discovery

The next important task after getting request definition is service matchmaking process or service discovery [8, 18,27]. In the matchmaking process, the system will search The Registry for composite service which has a similar graph structure with tenant’s request using similarity computation.

Say, for example, there is one composite service in the Registry as described in Figure 7. Similarity computation between graph structure in Figure 7 and Figure 8 begins with the similarity checking of nodes and edges. Dijkman in [24] used syntactic similarity on node labels using the Edit-Distance Algorithm.

In this study, we use Feature-based Similarity as seen in Equation (1) to compute similarity between nodes. This is because each node in the graph represents atomic web service, and each of them has features.

According to the result in Table 1, there are two similar nodes (indicated by the similarity value above threshold; threshold = 0.7) as follows.

1. `createReceiving()` similar with `provideReceiving()`
2. `checkQuantity()` similar with `checkQuantityDifference()`

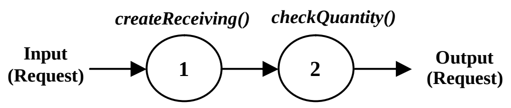
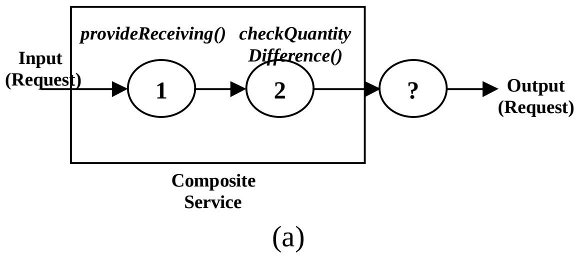


Figure 7. Example of Existing Composite Service.



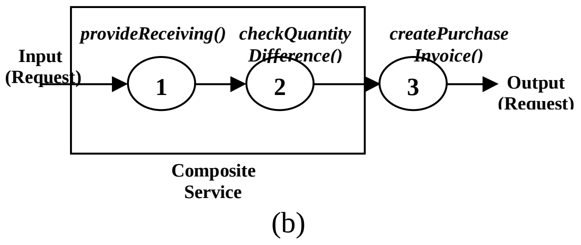


Figure 8. (a) Incomplete Composition Result; (b) Final Composition Result.

Thus, according to Graph-Edit Distance Algorithm, we got the following computation.

$$N1 = All\ nodes\ of\ Graph1 = 3$$

$$N2 = All\ nodes\ of\ Graph2 = 2$$

$$E1 = All\ edges\ of\ Graph1 = 4$$

$$E2 = All\ edges\ of\ Graph2 = 3$$

$$sn = All\ nodes - similar\ nodes = 5 - 4 = 1$$

$$se = All\ nodes - similar\ edges = 7 - 4 = 3$$

$$snv = \frac{|sn|}{N1+N2} = \frac{1}{5} = 0.2$$

$$sev = \frac{se}{E1+E2} = \frac{3}{7} = 0.42;$$

$$sbv = \frac{2\sum_{(n,m)\in M} 1 - Sim(n,m)}{|N1|+|N2|-|sn|} = \frac{2\left((1-0.93)+(1-0.86)\right)}{3+2-1} = \frac{0.42}{4} = 0.105$$

$$simgraph = 1 - avg(0.2 + 0.42 + 0.105) = 0.765$$

The result of Structure-based Similarity score is equal to 0.765 (above threshold). The result indicates the two graphs are similar. However, if we checked on the IOPE parameter, the output part of the found composite service still did not match with tenant’s output request. This condition is described in Figure 8(a).

In order to fulfill the request, there should be another service to close the gap. Therefore, to complementthe shortage,the algorithm will look for anothe rappropriate service using Feature-based Similarity where the target output is the output request from tenants, and target input isthe outputof the found composite service. The final resultof the service composition isshown in Figure 8(b).

The result shown in Figure 8(b) might be one possibility of composition plan. Each tenant's request can return several composition plans as a result of similarity computation. Each composition plan considered as a solution, if the average similarity score is above the threshold (threshold = 0.7).

Table 1. Feature-based Similarity Between Nodes.

No	Node/Features (Tenant)	Node/Features (Provider)	Feature Similarity
1	***createReceiving()*** **{(hasInputType,String),(hasOutputType,String), (hasInputName,purchaseOrderID), (hasOutputName,receivingID), (hasPrecondition,purchaseOrdered), (hasEffect,itemReceived)}**	***provideReceiving()*** **{(hasInputType,String),(hasOutputType,String), (hasInputName,purchaseOrderID), (hasOutputName,receivingID), (hasPrecondition,purchaseOrdered), (hasEffect,receivingCreated)}**	0.93
2	***checkQuantity()*** {(hasInputType,String), (hasOutputType,String), (hasInputName,purchaseOrderID), (hasOutputName,quantityDifference), (hasPrecondition,itemReceived), (hasEffect,quantityChecked)}	***checkQuantityDifference()*** {(hasInputType,String), (hasOutputType,String), (hasInputName,purchaseOrderID), (hasOutputName,quantityDifference), (hasPrecondition,receivingCreated), (hasEffect,qtyChecked)}	0.86
3	***createPurchaseInvoice()*** {(hasInputType,String), (hasOutputType,String), (hasInputName,purchaseOrderID), (hasOutputName,receivingID), (hasPrecondition,quantityChecked), (hasEffect,purchaseInvoiceCreated)}		0

Composing Complex Workflow

Example in the previous section shows only simple workflow consist of simple sequence. What if tenant, for example, request a more complex workflow consists not only sequence, but also conditional `ifThenElse` and parallel `splitJoin` execution. Figure 9 shows an example of a complex Order-to-Cash workflow in ERP.

As can be seen in Figure 9, the workflow begins when a customer order some products. Sales staff invoke *salesOrder* service to create sales order transaction. The process continue by invoking *checkStock* service for each product that is being ordered. After invoking *checkStock* service, there will be two possibilities. The product is in stock or out of stock. Therefore, the process continue with parallel excecution. For products that is in stock, the system will invoke *shipping* service, and at the same time, for products that is out of stock, the system will invoke *makeToOrder* service. The next process is to determine, whether the sales is having return or not. If it is true, the process continue to invoke *salesReturn*, *salesReturnInvoice* and *salesInvoice* consecutively, otherwise it invokes *salesInvoice* service. The process ends by invoking *cashbank* service.

Unfortunately, to find a similar composite service having similar workflow with the one shown in Fig. 9 are very difficult. The system will end up with no result because the similarity score is low. However, the search will be much easier if the target composition is simple enough.

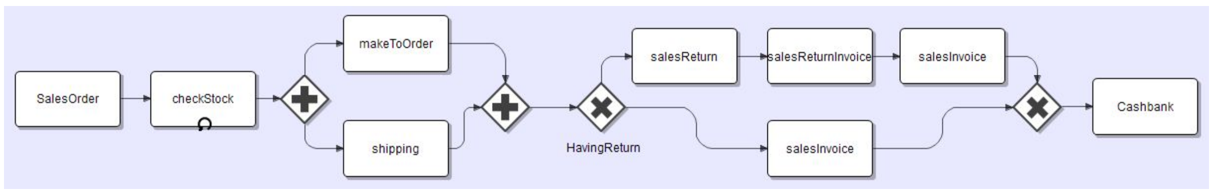


Figure 9. Complex Workflow.

Therefore, we developed our algorithm to addressed this problem as follows.

(i) Get the workflow
(ii) Break the workflow into sequence compositions
(iii) For each new sequence composition, do the following
(iv) If new sequence composition found a match composite service, use the composite service,
(v) Otherwise, make a new composition from atomic services in the Registry.
(vi) If all the sequences have been satisfied, compose the sequence together, otherwise back to step iii.

In our method, we break the structure into several sequence compositions to get the following benefits: (a) to make the searching and composing process much more easier because the target composition is simple; and (b) to gain flexibility in the composition process because new services can be added in the middle of the structure.

As in Figure 9, the workflow request can be break into 6 sequence composition as follows.

By breaking the workflow into sequences, the composition process will be much easier and flexible. For example, sequence Seq-4 in Table 2 as in Figure 10, might have a match with one composite service in the Registry. Thus, the composite service can be reusable.

Service composition for each sequence in Table 2, follow the steps and guide in the previous section for simple workflow. Each of the sequence will have their own IOPE definition to help the composition process.

For example, Seq-1, will have `Input` and `Precondition` definition from `salesOrder` service; and `Output` and `Effect` definition from `checkStock` service.

After all sequences have been satisfied, the next step will be combining all the sequences with the control construct as define in the workflow according to its structure.

Table 2. Breaking The Workflow.

Seq Composition	Services Sequence
Seq-1	salesOrder – checkStock
Seq-2	makeToOrder
Seq-3	shipping
Seq-4	salesReturn – salesReturnInvoice – salesInvoice
Seq-5	salesInvoice
Seq-6	Cashbank

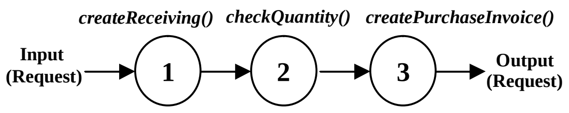


Figure 10. Sequence Seq-4.

RESULT AND DISCUSSION

The method proposed is tested in a service-based ERP application environment with 60 web services. The result indicated that the algorithm was able to compose web services that are similar to tenant's request specifications.

We use ROC (Receiver Operating Characteristics) classifier to classify the result of the composition discovery.

ROC pays attention on four possible conditions in the query result, namely, correct hit (True Positive/*TP*), correct rejection (True Negative/*TN*), incorrect hit (False Positive/*FP*) and incorrect rejection (False Negative/*FN*). The value of each condition can be obtain by manually observing the query result. For example, *TP* is the number of relevant result, while *FP* is not relevant. On the contrary, *TN* is the number of correct unreturned result, while FN is incorrect unreturned result.

Figure 11 describes ROC Curve, which is the plot of True Positive Rate (*TPR*) against False Positive Rate (*FPR*) of the result. *TPR* is the proportion of the relevant result returned by the system compare to all relevant result in the datasets.

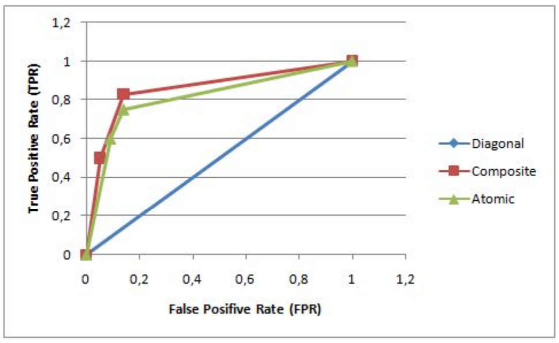


Figure 11. ROC Curve.

FPR is irrelevant result detected as irrelevant compare to all irrelevant service in the dataset. *TPR* and *FPR* can be determined by the equation as shown in Equation (2) and E quation (3).

$$TPR = \frac{TP}{TP+FN} \quad (2)$$

$$FPR = \frac{FP}{FP+TN} \quad (3)$$

The ROC Curve demonstrates several things. The closer the curve follows the left-hand border (above diagonal line), the more accurate the test. Additionally, the closer the curve comes to the 45-degree diagonal (under diagonal line), the less accurate the test. Based on curve in Figure 11, it can be seen that the matchmaking result for composite service is slightly better than atomic service.

The accuracy of the algorithm can be determined by formula as shown in Equation (4).

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (4)$$

By observing the result, we got the accuracy of this method is between 72% and 97%. The accuracy of 97% can be reached if all features of the request service were determined by tenant, and there were services with similar features in Provider's Registry. On the contrary, the accuracy of 72% was reached because tenant's determined only two out of four features, which are `input` and `output`.

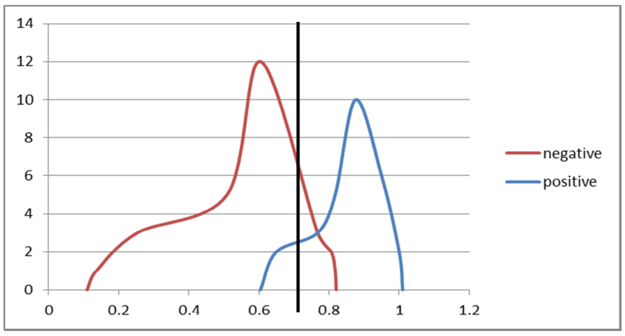


Figure 12. ROC Normal Curve.

Further analysis is to obtain the best threshold value in order to get the maximum accuracy. By adding more scenarios for 30 samples, we got normal distribution of the result as shown in Figure 12. The best threshold value should be in the intersection line between positive and negative result.

The left curve indicating True Negative result, and the right curve indicating True Positive result. The black-vertical line indicates the threshold value of 0.7.

Increasing the threshold value would result in fewer False Positive result. From the figure above, the point of curve intersection shows that the threshold should be 0.76 to get the maximum accuracy.

CONCLUSION AND FUTURE WORKS

In service-based application, there is a need, where tenants are able to construct their own business process according to their needs dynamically and semantically. This will make a new way for business to gain win-win solution between service provider and tenants.

In this paper, we proposed a method to make semantic composition using feature similarity and structural similarity.

Based on the composition formed, it can be concluded that the method used has been successfully compose web service that is similar to tenant's specification.

REFERENCES

[1] Q. Shao, "Towards Intelligent and Effective Multi-Tenancy SaaS", Ph.D. Dissertation, Arizona State University, Arizona, 2011.

[2] E.Sirin, B.Parsia, D.Wu, J.Hendler, and D. Nau, "HTN Planning for Web Service Composition Using SHOP2," *Web Semantics: Science, Services and Agents on*

the World Wide Web, vol. 1, pp.377-396. 2004.

[3] S.Narayanan and S.A. McIlraith, “Simulation, Verification, and Automated Composition of Web Services,” in *Proceedings of 11th International World Wide Web Conference (WWW-11)*, Honolulu, 2002.

[4] H.N.Talantikite, D. Aissani, and N.Boujdlida, “Semantics Annotation for Web Services Discovery and Composition,” *Computer Standards and Interface*, vol. 31, pp. 1108 – 1117, 2009.

[5] Q.Wang and P. Sheu, “Relational Service Composition,” in *Proceedings of International Conference on Semantic Computing (ICSC)*, Berkeley, 2009.

[6] A. Kunaefi and R. Sarno, “Ontology Mapping for ERP Business Process Variations,” in *Proceedings of Seminar Nasional Teknologi Informasi dan Multimedia,*Yogyakarta, 2013.

[7] A. Hijriani, R. Sarno, and R. Arinta, “Ontologi untuk Permodelan Service Level Agreement Web Service,” in *Proceedings of Seminar Nasional Teknologi Informasi dan Komputasi*, Bangkalan, 2012.

[8] Y. Anistyasari and R. Sarno, “Weighted Ontology for Subject Search in Learning Content Management System,” in *Proceedings of International Conference on Electrical Engineering and Informatics*, Bandung, 2011.

[9] A. Ankolekar, M. Burstein, J.R. Hobbs, O. Lassila, D. Martin, D. McDermott, S. A. McIlraith, S. Narayanan, M. Paolucci, T. Payne, and K. Sycara. “DAML-S: Web Service Description for the Semantic Web,” in *The Semantic Web-ISWC 2002: Proc. 1st International Semantic Web Conference*, Sardinia, 2002.

[10] A.S. Bilgin, and M.P. Singh, “A DAML-Based Repository for QoS-Aware Semantic Web Service Selection,” in *Proceedings of the IEEE International Conference on Web Service Selection*, Washington D.C, 2004.

[11] M. Klusch, B. Fries, and K. Sycara, “Automated Semantic Web Service Discovery with OWLS-MX,” in *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, New York, 2006.

[12] G. Meditskos and N. Bassiliades, “Structural and Role-oriented Web Service Discovery with Taxonomies in OWL-S,” *IEEE Transaction on Knowledge and Data Engineering*, vol. 22, pp. 278-290, 2010.

[13] A. Kunaefi, R. Sarno, “Orchestration of semantic web service using OWL-S for variations of ERP business process,” in *Proceedings of 2012 International Conference on Mathematics, Statistics and its Applications* (*ICMSA*), 2012

[14] J. Domingue, L. Cabral, F. Hakimpour, D. Sell, and E. Motta, “IRS-III: A Platform and Infrastructure for Creating WSMO-based Semantic Web Services,” in *Proceedings of the Workshop on WSMO Implementations (WIW’04)*, Frankfurt, 2004.

[15] D. Sell, F. Hakimpour, J. Domingue, E. Motta, and R.C.S. Pacheco, “Interactive Composition of WSMO-based Semantic Web Services in IRS-III,” in *Proceedings of 1st AKT Workshop of Semantic Web Services*, Milton Keynes, 2004.

[16] P. Bartalos, and M. Bielikova, “Adapting I/O Parameters of Web Services to Enhance Composition,” in *Proceedings of Fifth International Conference on Next Generation Web Services Practices*, Washington D.C, 2009.

[17] H. Wang, Z. Li and L. Fan, “An Unabridged Method Concerning Capability Matchmaking of Web Services,” in *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, Hongkong, 2006.

[18] R. Sarno, K. Ghozali, B.A. Nugroho, and A.Hijriani, “Semantic Match Making using Weighted Directed Acyclic Graph,” in *Proceedings of International Seminar on Applied Technology, Science, and Arts*, Surabaya, 2011.

[19] A. Malucelli, D. Palzer, and E. Oliveira, “Ontology-based Services to Help Solving The Heterogeneity Problem in e-Commerce Negotiations,” *Electronic Commerce Research and Applications*, vol 5, pp. 29-43. 2006.

[20] M. Ehrig, P. Haase, M. Hefke, and N. Stojanovic, “Similarity for Ontologies – A Comprehensive Framework,” in *Proceedings of* 13th*European Conference on Information Systems*,Regensburg, 2005.

[21] M. Horridge, S. Jupp, G. Moulton, A. Rector, and R. Stevens, “A Practical Guide To Building OWL Ontologies Using Protege 4 and CO-ODE Tools,” University of Manchester, Tech Report, 2007.

[22] Y. Ganjisaffar, H. Abolhassani, M. Neshati, and M. Jamali, “A Similarity Measure for OWL-S Annotated Web Service, in *Proceedings of 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, Hongkong, 2006.

[23] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, and K. Sycara, “OWL-S: Semantic Markup for Web Services,W3C Member Submission,” 2004.

[24] R. Dijkman, M. Dumas, B. Dongen, R. Kaarik, and J. Mendling, “Similarity of Business Process Models: Metrics and Evaluation,” *Journal of Informations Systems*, vol. 36, pp. 498-516, 2011.

[25] Y. Wang, W. Liu, and D.A. Bell, “A Structure-based Similarity Spreading Approach for Ontology Matching,” in *Proceedings of the 4th international conference on Scalable Uncertainty Management*, Toulouse, 2010.

[26] R. Sarno, ”Orchestrasi Web Services untuk Variasi Proses Bisnis ERP,” in *Proceedings of Seminar Nasional Sistem Informasi Indonesia*, Surabaya, 2012.

[27] Hermawan and R. Sarno, “Developing Distributed System with Service Resource Oriented Architecture,” *TELKOMNIKA International Journal*, vol.10, pp. 389-399, 2012.