

# Dynamic Structure Mechanism Based on Lightweight Relevancy for Web Services Directory

S.M.F.D Syed Mustapha  
Asian Research Center  
British Telecom Multimedia Sdn Bhd  
Cyberjaya, 63000  
Selangor, MALAYSIA

*Abstract - The task of composing Web services workflow can be unwieldy for a huge size Web services network. The challenge is even intensified when dealing with continuous changes of the business practices, processes and transactions. The dynamic business environment compels the Web service providers to do high maintenance for the directory update to ensure validity, consistency and seamless service searching. In order to build Web service directories that are amenable to the business changes, this paper showed a novel solution using dynamic structure mechanism based on lightweight relevancy for the Web service directory. The mechanism works on the two main techniques: lightweight-relevancy modelling and forward-chaining monitoring. Lightweight-relevancy modelling simplifies the semantic link in order to reduce the searching effort; forward-chaining monitoring that maintains the relevancy and consistency. The dynamic structure mechanism suitably positioned as the interface layer between the discovery engine of the Web service agency and the Web service directories.*

**Keywords:** UDDI registry, lightweight relevancy, Web service discovery, intelligent technique in Web services

## 1.0 Introduction

Automated Web service composer builds the required components of services based on the business activities and business workflow that are defined before or during the execution. In the mobile Web service environment, the latter would be the required setting since the types of transaction and activities being involved are necessarily not known a priori. The composition is more complex when dealing with different nature of QoS in the business and also the variants of the functional and non-functional aspects of the Web services. In our perspective, the Web services composition needs to consider beyond the traditional way of composing the required Web service components which was done in a static manner. Some add-on features that Web service composer should be able to do before the composition are such as substituting, analyzing, ranking and selecting the appropriate service based on certain prescribed factors. For example, the Web services  $WS_A$  and  $WS_B$  offer services for ticket reservation but  $WS_B$  is ranked higher due to its transaction simplicity; or  $WS_B$  is frequently selected for its guaranteed service availability; or one can be considered as alternative service of the other when either is not reachable.

In order to enable this to happen, the technology to compose the available Web services from a directory is needed. The current researches have made attempt in two different ways, manual and automated composition. In the former, an interface is provided for the service provider to choose among the available services from the Web service directories at the design time. The latter is more complex that prior to the composition, a semantic understanding of the Web service functionality and also the business logic have to be built. This process is rather expensive and the development is time-consuming. The task becomes more challenging when business rules and business models change over time and therefore, the semantic modelling has to be modified in tandem to business processes. A frequent update to the information in the Web service registry (in general, the term 'directory' is used) may have an adverse impact to the discovery process and accuracy which is an unacceptable notion for efficiency reasons. In order to address the problem, we introduce a Lightweight Relevancy Mechanism (thereafter called LwRM) in building the dynamic structure of Web services directory.

Section 2 describes the concept of Lightweight Relevancy Mechanism and its benefit, section 3 explains the features of the relevancy, section 4 presents the forward chaining algorithm for monitoring relevance network and the conclusion and future work are given in the section 5.

## 2.0 Lightweight Relevancy Mechanism (LwRM)

The process of finding the relevancy between Web services that are semantically modelled involves a massive calculation using sophisticated measuring similarities function for some complex variables. The approach is most appropriate for a static Web services workflow environment but impractical when considering a weighty Web services network. LwRM is a simplified version of searching Web services and it resides at intermediary layer between the Web service directory and the traditional Web service discovery engine. The purpose is to simplify the discovery or searching effort by skimming the most likely relevant Web services component from the entire directory database and perform a preliminary selection based on certain relevancy criteria. The simplified skimming process cut off the unnecessary searching detail and consequently reduces the computational time.

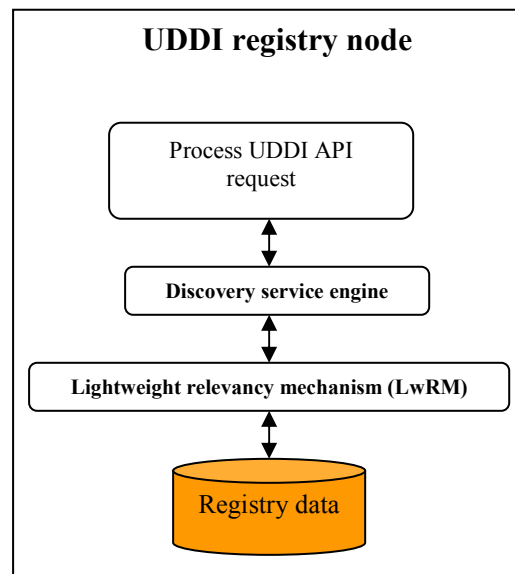


Figure 1. Incorporating lightweight relevancy mechanism in UDDI registry node  
(Adapted from [11])

Figure 1 shows the internal scheme of the UDDI<sup>1</sup> registry node where LwRM is positioned between the discovery service engine and the registry data (service directory). In the traditional approach, the UDDI request which is received by the discovery service engine will be processed to retrieve the relevant Web services from the registry data. The structure of the registry data is becoming more complex with the integration of semantic modelling of the Web services. As a result, the searching effort can be exhaustive and time-consuming for huge set of directory services. LwRM is incorporated to execute three main tasks which shall address the problems:

- a. to build a dynamic structure automatically that links Web services components based on lightweight-relevancy criteria. The lightweight-relevancy criteria will be defined in Section III.
- b. to monitor and update the relevancy network of the Web services directories using forward-chaining monitoring technique periodically. The forward-chaining algorithm is discussed in Section IV.
- c. to retrieve the relevant Web service components based on the request to feed back the responses to the discovery service engine.

<sup>1</sup> Universal Description Discovery and Integration

In this paper, UDDI is used as an illustration as it is one of the popular Web service directory services. Other examples of service directories currently adopted by the industries are X.500, LDAP, Cos Trader from OMG (Object Management Group), FIPA directories and JINI could also adopt this mechanism. In the following section, we define the relevancy issues and formalised the idea for generality purposes.

### 3.0 Relevancy issues in the Web service searching

LwRM builds the relevance network of Web service components. These components are connected as a single layer (non-hierarchical) and undirected. The reason for this structure is the simplistic of the computation that LwRM undertakes in order to reduce computational time and cost. The connection also needs to be simple as the relevancy is continuously monitored and the modification to the connection is done dynamically over the time.

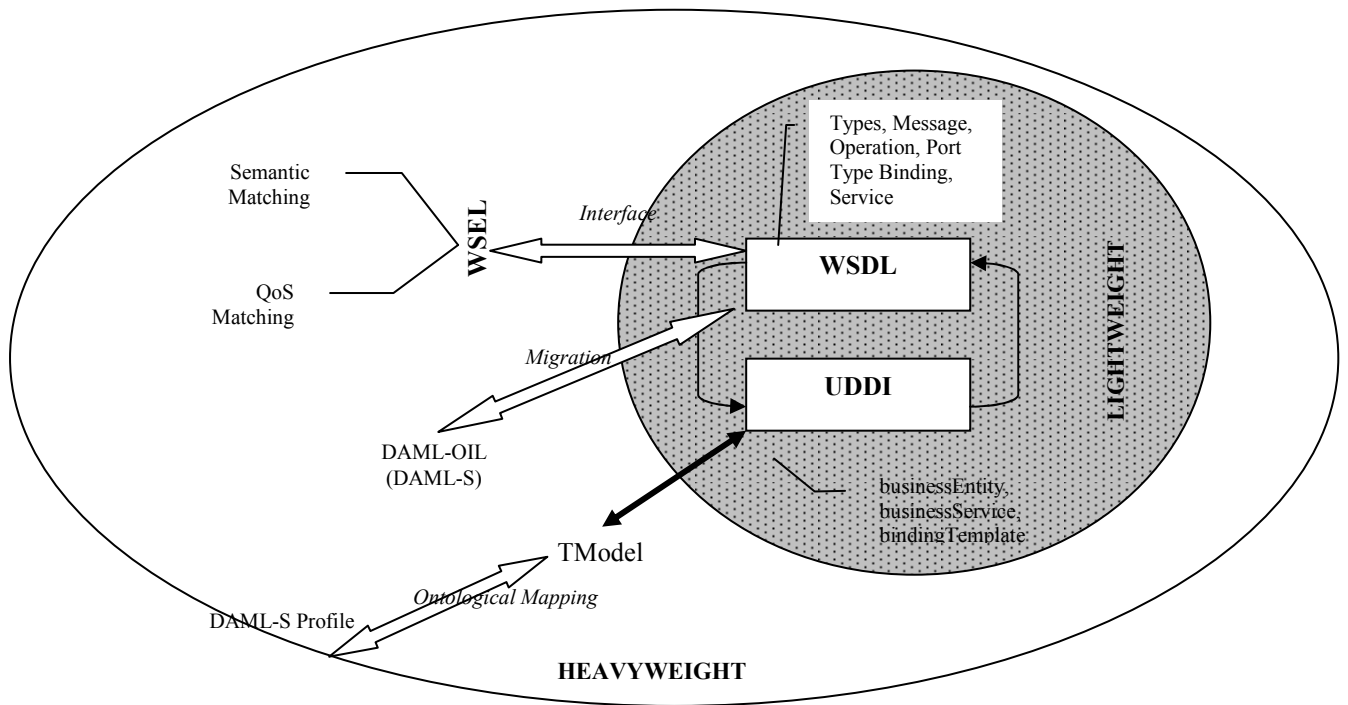


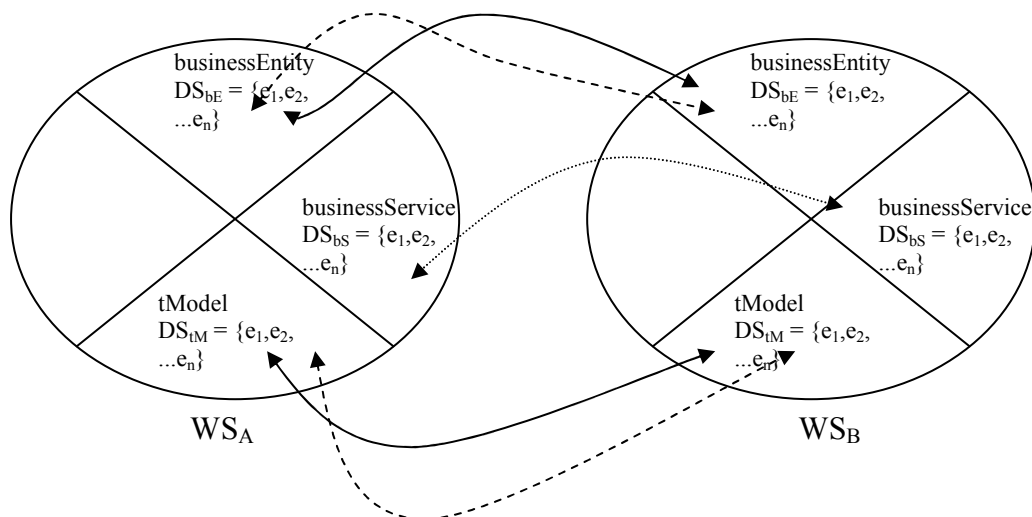
Fig 1. Classification of Web service registry modelling – Lightweight and Heavyweight

There are several approaches in modelling the Web service registry and they can be classified as lightweight and heavyweight as shown in Fig. 1. The UDDI is an industrial effort to standardize and integrate the Web service directory. The UDDI and other directory services are developed progressively towards supporting the semantic meaning of the functional and non-functional aspects of the Web services and consequently increase the matching and selection accuracy. Research has shown that there are two approaches being adopted to achieve this objective. First, by incorporating the semantics to the Web services standard and secondly, by isolating the semantic processor that creates independency to the WS standard description. Sivashanmugam, et al [1,2] disintegrated the extension of semantic modelling and processing by creating an interface thru WSEL (Web Service End-point Language). The semantics and the QoS (Quality of Service) features are modelled as separate entities such that the Web service provider has a flexible option to publish with or without them. In contrary, Paolucci, et al [3,4] transform the WSDL features to DAML-S representation that provides a direct communication between the semantics and matching engine. Both of these approaches are classified as heavyweight model as they need a deeper level of searching and reasoning and require a more complex development and programming effort. While the intention is to improve the matching at semantic level, the adverse effect could be high maintenance cost for future changes on business aspects and also the

standard architecture of the registry. For example, the evolutionary UDDI proposed the distributed environment and multiple registries as opposed to the earlier Universal Business Registry [12].

The relevancy criteria model in LwRM determines the types of connection between the two Web service components. Two Web service components can have more than one type of connection depending on the relevancy criteria fulfilled by both of them. Since it is designed to be a lightweight model, massive computation to perform similarity measures, ontological mapping, multi-level semantic features is unnecessary. Each type of the connection represents a criterion in which the similarity is measured on a keyword basis. The LwRM is argued to be useful that it keeps the relevancy links of all the Web services components in a loose manner with superficial matching criteria. Fig 2 illustrates a simple example of the idea of two Web services using UDDI structure as an illustration.

$WS_A$  and  $WS_B$  are two Web services in which the association links are modelled using LwRM. The data structure (DS)<sup>2</sup> comprises of elements for each component in the registry; an example for businessEntity  $DS_{bE} = \{e_1, e_2, \dots, e_n\}$ . The elements are service-provider-defined and these reflect the identity and functionality of the Web service, for example in this case (businessEntity) there could be two elements, namely, the *name* and *description*. The matching is performed on the elements of the same type. In an ideal case which is rare is the exact match, where the keywords of the element on both sides are equal/very close. An element from a DS in  $WS_A$  subsumes to the same element type of a DS in  $WS_B$  if all of the keywords of the former are contained in the bag of words in the latter. If neither of the elements from both sides subsumes, then the only possible match is a partial match. When none of these applies, the matching does not exist.



Legend:

- exact match
- - - - - subsume match
- ..... partial match

Fig. 2 An illustration of Web services  $WS_A$  and  $WS_B$  components that have relevancy link using LwRM

The matching uses superficial data to build the connection among the Web service components. These data should appear as basic input provided by any service provider according to standard specification (e.g. OASIS, W3C). The connection established in the LwRM is subject to the following understanding about the Web service relevancy link:

- a. epitomic – the data is brief that can be processed through a simple text parsing technique and keyword matching. In other words, it is an abstract representation that omits detail processing such as the numerical encoding of service description [5,6] and the negotiation of case matching/searching [7].
- b. multivariate – the specification of each Web service in a registry differs that some fields are not obligatory to all. This makes, in some cases, matching of the same type field not to be possible. Thus, LwRM should allow uncomparable field.

<sup>2</sup> The term data structure to be consistent with UDDI [11]

The Lightweight Relevancy Mechanism is modelled based on the above understanding using the three types of matching. Thus, the relevancy,  $\mathfrak{R}$ , is calculated as follows,

$$\mathfrak{R} = \sum_i^M \sum_j^N \text{match} \left[ \left( e_j^i \right)_{WS_A}, \left( e_j^i \right)_{WS_B} \right]$$

where  $M$  is the number of data structure in the registry,  $N$  is the number of elements in the data structure. The  $WS_A$  and  $WS_B$  are the two illustrated Web services described in Fig 2 and  $e$  represents the element of the respective DS (superscripted with  $i$ ) and its series (subscripted with  $j$ ). The matching function is the dominant factor in determining the relevancy value. LwRM proposes the three types of matching condition: exact match, subsume match and partial match. In qualitative terms, the exact match has the highest relevance value followed by the subsume match and finally the partial match. Numerically it can be translated as follows:

$$\text{match}(a, b) = \begin{cases} 1.0 & \text{exact, } \frac{n(f(l_a) \cap f(l_b))}{n(f(l_a))} \quad \forall l \in a, b \\ 0.5 & \text{subsume, } \frac{n(f(l_a) \subset f(l_b))}{n(f(l_a) \cup f(l_b))} \quad \forall l \in a, b \\ 0.1 & \text{partial, } \frac{n(f(l_a) \Leftrightarrow f(l_b))}{n(f(l_a) \cup f(l_b))} \quad \forall l \in a, b \end{cases}$$

where  $l$  is the sub-element (for example the letters of a word/description),  $n$  is the cardinality and  $f$  is a function that maps onto the non-empty set of sub-elements of  $l$ .

The LwRM maintains a brief connection of each element in every data structure such that it is prepared to alter the connection type as recommended by the monitoring process as explained in the following section.

## 4.0 Monitoring based on forward chaining algorithm

Forward chaining had been used as a technique for monitoring dynamic system [8]. Recent works had shown its usefulness in the Web service composition and directory service integration [9,10]. Forward chaining is chosen for this application due to large possible events that may occur and they are unpredictable. The purpose of the monitoring is to detect changes in any part of the directory services such that if that happens, the relevancy links described in Section 3.0 (Fig. 2) have to be reconstructed. The changes such as adding new record to registry, modifying the business description, removing the data structure or elements will need a reconstruction to the links.

Forward chaining is an inference technique commonly represented in a rule-based form. For generalization purpose, the forward chaining is described in algorithmic way as follows:

$$\begin{aligned} R &= \{r_1, r_2, \dots, r_N\} \\ DS &= \{ds_1, ds_2, \dots, ds_N\} \\ E &= \{e_1, e_2, \dots, e_N\} \\ EV &= \{M, D, I\} \\ A &= \{FR, PR\} \end{aligned}$$

1.  $EV = (M \mid D \mid I) \rightarrow \varphi$
2.  $\text{Detect}(\varphi) = \{R \mid DS \mid E\} \rightarrow \{\text{true} \mid \text{false}\}$
3. if  $\text{Detect}(\varphi) \leftarrow \text{true}$ 
  - a.  $(\varphi) = R$ , then perform FR
  - b.  $(\varphi) = DS$ , then perform FR
  - c.  $(\varphi) = E$ , then perform PR
4. Repeat 1,  $\forall \varphi \in EV$

The symbols R – registry, DS – data structure, E – elements in DS, EV – events (M-modification, D – deletion and I – insertion) that can be applied on (R, DS, E) and A – recommended action (FR – full reconstruction, PR – partial reconstruction). Since forward chaining does not start with a goal, it is data-driven or in this case, event-driven. To start, EV assigns its first value to  $\varnothing$  (say M, to check any occurrence of modification). The detection process is applied across the registry, data structure and elements and returns true or false. If true value is obtained, there are two ways to react to the detection of these changes namely, a reconstruction of the entire relevancy network (FR) or modification (PR) to the relevant links that pointed to the affected part. The former is required for R and DS as they are massively linked compare to the individual elements.

It is important to point out that the forward chaining algorithm given above is a straightforward example in the case of UDDI format and appropriate modification is needed to tailor for different directory registries.

## 5.0 Conclusion and future work

Registry is a growing research interest since it is the inception point of acquiring available Web services. The structure in the registry is important to ensure searching of relevant Web services can be done seamlessly. It is assumed that fast growing number of Web services will increase the searching time exponentially. Therefore, the structure of the directory service is an important factor to ensure timely retrieval result. The recent implementation of semantics in service directory complicates the issue for fast searching. The past approaches discussed in this paper showed the pros and cons in terms of retrieval accuracy and the retrieval turnaround. It is argued also that incorporating semantics in the registry creates inconsistencies of directory structure among Web services and inappropriate to apply standard searching mechanism. On the other hand, a deep semantic model built externally requires interface language to bridge the directory service. In order to resolve this problem, the approach taken is to create a middle layer that keeps a loose network connection between Web services based on lightweight relevancy criteria. The construction of the network connection uses simple matching strategy. The advantage is that it reduces the complexity and time-consuming development time as the criteria is greatly simplified. The philosophy behind this is to allow a fast browsing to search for a potential relevant Web services. Forward chaining algorithm is used to monitor the changes of the data and information and reconstruct the connection appropriately.

LwRM has strength in terms of relevancy browsing but it may sacrifice the accuracy in order to gain the searching time and also indexing the directory service. It does not need to standardize the registry format due to its simplified matching criteria. Nevertheless, it will be an interesting future work to look at the accuracy factor while maintaining its innate strength. The semantic information will improve the accuracy and at the same time provide a meaningful connection between the Web services. Another direction to improve the work is on the forward chaining algorithm where the present algorithm has no means to calculate intelligently the impact factor of the changes that took place.

## 6.0 References

- [1] Framework for Semantic Web Process Composition. International Journal of Electronic Commerce, Winter, 2004-5, Vol. 9(2), pp 71-106.
- [2] Sivashanmugam, K., Verna, K., Sheth, A.P. and Miller, J.A. Adding Semantics to Web Services Standard. In 1st International Conference on Web Services (ICWS'03), pages 395-401, June 2003.
- [3] Paolucci, M., Kawamura, T., Payne, T.R., Sycara, K. Importing the Semantic Web in UDDI, In CAiSE 2002.
- [4] Paolucci, M., Kawamura, T., Payne, T.R., Sycara, K. Semantic matching of Web services capabilities. In I. Horrocks and J. Handler, editors, 1st Int. Semantic Web Conference (ISWC), pages 333--347. Springer Verlag, 2002.
- [5] Constantinescu, I., Binder, W. and Faltings, B. Directory Services for Incremental Service Integration. The Semantic Web: Research and Applications: First European Semantic Web Symposium, ESWS 2004 Heraklion, Crete, Greece, May 10-12, 2004.
- [6] Constantinescu, I. and Faltings, B. Efficient Matchmaking and Directory Services p. 75, IEEE/WIC International Conference on Web Intelligence (WI'03), 2003.

- [7] B. Limthanmaphon, Z. Zhang, and Y. Zhang, Adaptive Case-Based Reasoning Systems for E-Commerce, in Proceedings International Conference on Intelligent Information Technology, Beijing Sep 22-25,2002.
- [8] Blum, A.L. and Furst, M. L. Fast planning through planning graph analysis. Artificial Intelligence, 90(1-2): 281-300, 1997.
- [9] Thakkar, S., et al. Dynamically Composing Web Services from On-line Sources. In Proceeding of 2002 AAAI Workshop on Intelligent Service Integration. 2002. Edmonton, Alberta, Canada
- [10] Constantinescu, I., Faltings, B., Binder, W. Large Scale, Type-Compatible Service Composition, p. 506, IEEE International Conference on Web Services (ICWS'04), 2004.
- [11] Bellwood, T. Understanding UDDI, IBM developerWorks Journal, May 2002. Available at <http://www-128.ibm.com/developerworks/library/ws-featuddi/>
- [12] The Evolution of UDDI, UDDI.org White Paper, 2002  
[http://www.uddi.org/pubs/the\\_evolution\\_of\\_uddi\\_20020719.pdf](http://www.uddi.org/pubs/the_evolution_of_uddi_20020719.pdf)