RC23133 (W0403-026) March 2, 2004
Computer Science

IBM Research Report

Parameterized Semantic Matchmaking for Workflow Composition

Prashant Doshi
Dept. of Computer Science
Univ. of Illinois at Chicago
Chicago, IL 60612

Richard Goodwin, Rama Akkiraju
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Sascha Roeder
IBM
Hechstheimer Str. 2
Mainz 55131
Germany

Parameterized Semantic Matchmaking for Workflow Composition

Paper ID: 195

Prashant Doshi
Dept. of Computer Science
Univ. of Illinois at Chicago
pdoshi@cs.uic.edu

Richard Goodwin and Rama Akkiraju
IBM T. J. Watson Research Center
Hawthorne, NY
{rgoodwin,akkiraju}@us.ibm.com

Sascha Roeder
IBM
Mainz, Germany
sroeder@de.ibm.com

***Abstract*— Workflow compositions in the context of Web services represent an important application of semantic matchmaking. Existing matchmaking frameworks utilizing a strict capability-based matchmaking prove to be inadequate for this application. To address these limitations, we present a parameterized semantic matchmaking framework that exhibits a customizable matchmaking behavior. Using an example scenario, we argue that such customization is essential for application in workflows, and demonstrate the wider applicability of our matchmaking algorithm. Additionally, we outline an industrial application of our technique.**

I. Introduction

Workflow composition in the context of Web services involves discovering and binding to Web services that collectively implement the required process functionality. The composition may either be static or dynamic depending on the process environment. Highly dynamic environments require the workflow to adapt continuously, during which new Web services may be discovered and invoked. Furthermore, the flows may either be intra-organizational or cross-organizational. Intra-organizational workflows connect Web services within the same organization. Cross-organization workflows span multiple organizations and tie together the Web services of the participating organizations.

In workflow compositions Web services serve the role of building blocks. The loose coupling delivered by Web services facilitate automatic generation of dynamic workflows, provided technology to discover the required Web services exists. Early efforts in the area of Web services discovery have resulted in preliminary *matchmaking* frameworks that attempt to match a service request with a service advertisement. Web services annotated with semantic information[1] enable semantic matching that is more robust than pure syntactic matching. Inclusion of semantic information in the matching process is a proven technique in the Information Retrieval domain [1]for retrieving more useful search results than simple syntactic comparisons.

The primary contribution of our paper is in presenting a semantic matchmaking framework that is tailored towards workflow compositions. Specifically, our matchmaking framework permits complete customization of the matching process by allowing external specification of the matchmaking criteria. In contrast, existing frameworks exhibit an immutable matching process and use only service capability (input, output, preconditions, and, effects) as the criteria for a match. It is our hypothesis that the criteria utilized for matching services is subjective to the matchmaking application. A simple example that motivates our hypothesis is in composing cross-organizational workflows where quality of the process must be maintained. This requires the target Web service to have a predefined *quality of service* in addition to the required capability, without which the match will fail. Thus, strict capability-based matchmaking proves to be inadequate in this scenario. Furthermore, as a step towards increasing the ”hit” ratio of successful matches, we suggest utilization of backward-chaining based composition of Web services as part of our matchmaking framework. Hence, our matchmaking framework uses a two-step approach towards service discovery. Firstly, a direct match between service request and some advertisement is attempted. If this proves impossible, a subset of the advertisements are composed into an aggregate service using backward chaining that satisfies the request. We demonstrate the applicability of our approach by discussing its potential use in an industrial setting. In particular, business process integration and management (BPIM) appears to be a field where our matchmaking framework may find good use.

The rest of this paper is structured in the following manner. In the next section, we briefly discuss the composition of workflows in the context of an example scenario. We then present our semantic matchmaking framework. An application of our technique follows.

II. Workflow Composition

The prevalent technique in workflow composition follows a manual and tedious approach involving assimilation of varied process design and vendor specifications, and writing vast amounts of code that produces a tight inflexible coupling between processes. We believe that the distributed computing platform provided by Web services possesses the potential to automate this technique to a great extent.

[1]Web services annotated with semantic information are popularly labeled as ”Semantic Web Services”

Limited prior work exists in the area of applying Web services towards workflow compositions. Laukkanen et al [2] explore the role of Semantic Web services and matchmaking in composing workflows. Specifically, they outline two cases of workflow compositions involving Web services. In the first case, the flow is pre-defined but some component Web service fails; and in the second case, a new flow must be established. Both these cases place heavy emphasis on discovering target Web services.

In order to elucidate the composition of workflows using Web services we present an example cross-organizational workflow. The workflow describes a simple supply-chain process, where merchandise is ordered by a retailer from a manufacturer who in turn orders parts from its supplier to satisfy the order.

Definition 1 (Example scenario): A manufacturer receives an order to deliver some merchandise to a retailer. The manufacturer on checking its inventory may realize a deficit in parts to build the merchandise. The manufacturer then places an order of parts with a supplier. Manufacturing company policy requires all transactions to occur with a pre-defined quality of service. Additionally, to curtail delivery times, the supplier must be within a pre-defined geographical radius from the manufacturer.

We illustrate the above scenario using a UML Activity Diagram in Fig 1. The diagram serves a two-fold purpose: It clearly demonstrates the role of Web services as building blocks in future workflows; and it emphasizes the need for a customizable matchmaking technology for constructing the workflow.

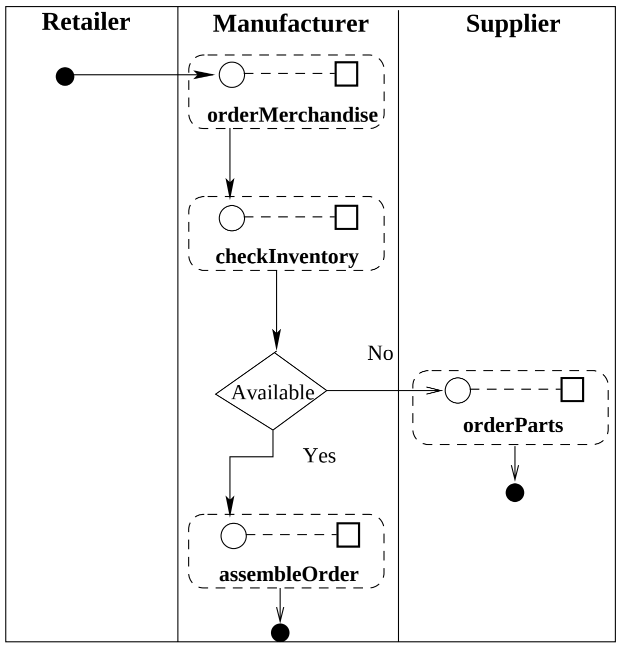

Fig. 1. Each activity consists of a call to the matchmaker (denoted by a circle) followed by discovering and binding to the target Web service (denoted by a square).

III. Matchmaking Framework

The design of our matchmaking framework is predicated on the need for a customizable matching process that permits the programmer to control the attributes matched, the order in which they are compared, as well as the closeness of matching desired. In addition to making the matching engine light, such an approach leads to the development of a generic matchmaking framework that may be utilized for a variety of matching tasks.

Features	**Matchmaking Frameworks**		
	MATCHMAKER	RACER	PSMF
Match Criteria	Capability only	Capability only	All attributes
Markup Lang.	DAML-S, UDDI	DAML-S	DAML-S , WSDL, UDDI
Customizable	No	No	Completely
Speed	Moderate due to various extra filters	Fast	Fast
Performance	Excellent	Good	Good
Target Application	E-commerce	E-commerce	Workflow composition

TABLE I

A SUBJECTIVE ASSESSMENT OF VARIOUS MATCHMAKING FRAMEWORKS

In Table I we compare our matchmaking effort (**PSMF** – Parameterized Semantic Matching Framework) with other related frameworks[2]. One of the earliest matchmaking engine is Sycara et al' MatchMaker [3] that is available on the Web as a service. In addition to utilizing a capability-based semantic match, the engine also uses various other IR-based filters thereby reducing the number of false positives at the expense of speed of matching. Another related effort is Racer [4], that focuses solely on a service capability-based semantic match for application in e-commerce systems. In contrast, our matchmaking framework provides a high degree of customizability, permitting the service requester who is most knowlegeable about its needs, to select the desired match criteria. This translates into a highly flexible, generative matchmaking behavior that is well suited to various workflow compositions.

A. Matching Algorithm

Service matching may be defined as the process of discovering a service advertisement that *sufficiently* satisfies a service request. Before we formalize the concept of *sufficiency*, we require the definition of a *service* and other service-specific notions from the perspective of matchmaking.

Definition 2 (Service): A service, $\mathcal{S}$, is defined simply as a collection of attributes that describe the service. Let $\mathcal{S}.\mathcal{A}$ denote the set of attributes of service, $\mathcal{S}$, and $\mathcal{S}.\mathcal{A}_i$ denote each member of this set. Let $\mathcal{S}.\mathcal{N}$ denote the cardinality of this set.

[2]Lack of benchmarks and standardized Web services repositories preclude an objective performance assessment.

Example 1: The manufacturer provides a Web service, **orderMerchandise**, that is defined by the following attributes: service category, input, output, preconditions, postconditions, quality of service, and geographical location.

Definition 3 (Service Capability): The capability of a service, $\mathcal{S}.\mathcal{C}$, is a subset of service attributes ($\mathcal{S}.\mathcal{C} \subseteq \mathcal{S}.\mathcal{A}$), and includes only those that directly relate to its working.

Example 2: The capability of **orderMerchandise** is, $\mathcal{S}.\mathcal{C}$ = {input, output, preconditions, postconditions}

Definition 4 (Service Property): The property of a service, $\mathcal{S}.\mathcal{P}$, is a subset of service attributes ($\mathcal{S}.\mathcal{P} \subseteq \mathcal{S}.\mathcal{A}$), and includes all attributes other than those included in service capability. Formally, $\mathcal{S}.\mathcal{P} = \mathcal{S}.\mathcal{A} - \mathcal{S}.\mathcal{C}$.

Example 3: The property of **orderMerchandise** is, $\mathcal{S}.\mathcal{P}$ = {service category, quality of service, geographical location}

A semantic match between two entities frequently involves a similarity measure. The similarity measure quantifies the semantic distance between the two entities participating in the match.

Definition 5 (Similarity measure): The similarity measure, μ, of two service attributes is a mapping that measures the semantic distance between the conceptual annotations associated with the service attributes. Mathematically,

$$\mu : \mathcal{A} \times \mathcal{A} \rightarrow \begin{array}{l}\{\text{Exact,Plug-in, Subsumption, Container,} \\ \text{Part-of, Disjoint}\}\end{array}$$

where $\mathcal{A}$ is the set of all possible attributes.

If the two conceptual annotations are syntactically identical, the mapping is called an **Exact** map. If the first conceptual annotation is specialized by the second, the mapping is called a **Plug-in** map. If the first conceptual annotation specializes the second, the mapping is called a **Subsumption** map. If the first conceptual annotation contains the second, the mapping is called a **Container** map, and if the first conceptual annotation is a part of the second, the mapping is called a **Part-of** map. Otherwise, the mapping is called a **Disjoint** map.

A preferential total order may now be established on the above mentioned similarity maps. We give this preferential order below.

Definition 6 (Similarity measure preference): Preference amongst similarity measures is governed by the following strict total order:

Exact $\succ$ **Plug-in** $\succ$ **Subsumption** $\succ$ **Container** $\succ$ **Part-of** $\succ$ **Disjoint**

Here, a $\succ$ b means that a is preferred over b.

Other matchmaking frameworks [3] utilize an idea similar to μ, but label it as a "degree of match".

We are now ready to define a *sufficient* match between services.

Definition 7 (Sufficient service match): Let $\mathcal{S}^{\mathcal{R}}$ be the service that is requested, and $\mathcal{S}^{\mathcal{A}}$ be the service that is advertised. Let $\mathcal{S}^{\mathcal{R}}.\mathcal{A}$ be the set of attributes to be utilized for matching. $\mathcal{S}^{\mathcal{R}}.\mathcal{A}$ may include both service capability as well as service properties. Let μ_i be the desired similarity measure for each service attribute $\mathcal{S}^{\mathcal{R}}.\mathcal{A}_i$. A sufficient match exists between $\mathcal{S}^{\mathcal{R}}$ and $\mathcal{S}^{\mathcal{A}}$ if for every attribute of $\mathcal{S}^{\mathcal{R}}$ there exists an identical attribute of $\mathcal{S}^{\mathcal{A}}$ and the values of the attributes satisfy the desired similarity measure. Formally,

$$\begin{array}{l}\forall_i \exists_j \ \ (\mathcal{S}^{\mathcal{R}}.\mathcal{A}_i = \mathcal{S}^{\mathcal{A}}.\mathcal{A}_j) \ \wedge \ \mu(\mathcal{S}^{\mathcal{R}}.\mathcal{A}_i, \mathcal{S}^{\mathcal{A}}.\mathcal{A}_j) \succeq \mu_i \\ \Rightarrow \text{SuffMatch}(\mathcal{S}^{\mathcal{R}}, \mathcal{S}^{\mathcal{A}}) \quad 1 \leq i \leq \mathcal{S}^{\mathcal{R}}.\mathcal{N}\end{array} \tag{1}$$

Sentence 1 parameterizes the desired similarity measure, μ_i, for each attribute, $\mathcal{S}^{\mathcal{R}}.\mathcal{A}_i$. Further customization of the matching process is carried out by specifying which attributes of the service request should be utilized during the matching process, and the order in which the attributes must be considered for comparison. An added advantage of this customization ability is that a single service request may be reused for multiple matching tasks thereby generating different matches. In order to permit this customization, we define a Criteria Table that serves as a parameter to the matching process.

Definition 8 (Criteria Table): A Criteria Table, $\mathcal{C}$, is a relation consisting of two attributes, $\mathcal{C}.\mathcal{A}$ and $\mathcal{C}.\mathcal{M}$. $\mathcal{C}.\mathcal{A}$ describes the service attribute to be compared, and $\mathcal{C}.\mathcal{M}$ gives the least preferred similarity measure for that attribute. Let $\mathcal{C}.\mathcal{A}_i$ and $\mathcal{C}.\mathcal{M}_i$ denote the service attribute value and the desired measure in the i^{th} tuple of the relation. $\mathcal{C}.\eta$ denotes the total number of tuples in $\mathcal{C}$.

Example 4: An example Criteria Table is shown in Table II.

Criteria Table ($\mathcal{C}$)	
$\mathcal{C}.\mathcal{A}$	$\mathcal{C}.\mathcal{M}$
service category	Subsumes
output	Exact
quality of service	Plug-in

TABLE II
AN EXAMPLE CRITERIA TABLE

In light of Definition 8, the sufficiency condition (1) can be rewritten. A sufficient match exists between $\mathcal{S}^{\mathcal{R}}$ and $\mathcal{S}^{\mathcal{A}}$ if for every attribute in $\mathcal{C}.\mathcal{A}$ there exists an identical attribute of $\mathcal{S}^{\mathcal{R}}$ and $\mathcal{S}^{\mathcal{A}}$ and the values of the attributes satisfy the desired similarity measure as specified in $\mathcal{C}.\mathcal{M}$.

$$\begin{array}{l}\forall_i \exists_{j,k} \ \ (\mathcal{C}.\mathcal{A}_i = \mathcal{S}^{\mathcal{R}}.\mathcal{A}_j = \mathcal{S}^{\mathcal{A}}.\mathcal{A}_k) \ \wedge \ \mu(\mathcal{S}^{\mathcal{R}}.\mathcal{A}_j, \mathcal{S}^{\mathcal{A}}.\mathcal{A}_k) \\ \succeq \mathcal{C}.\mathcal{M}_i \Rightarrow \text{SuffMatch}(\mathcal{S}^{\mathcal{R}}, \mathcal{S}^{\mathcal{A}}) \quad 1 \leq i \leq \mathcal{C}.\eta\end{array} \tag{2}$$

Our matching algorithm, Fig 2, follows directly from Sentence 2. Our algorithm is in part motivated by a similar semantic matching algorithm given in [5] . However, the latter algorithm permits only a service capability-based match. In contrast, our algorithm places no restrictions on the attributes used during matching.

Computational Complexity To understand the asymptotic complexity of our matchmaking algorithm, we observe that the process of computing μ is the most "expensive" step of the algorithm. Typically, $\mathcal{S}^{\mathcal{A}}.\mathcal{N} >> \mathcal{S}^{\mathcal{R}}.\mathcal{N}$, hence complexity of the first outer $while$ loop is $\mathcal{O}(\mathcal{C}.\eta \times \mathcal{S}^{\mathcal{A}}.\mathcal{N})$. Our approach to infer $\mu(\cdot,\cdot)$ is to parse ontological pieces of information

```
serviceMatch($\mathcal{S}^{\mathcal{R}}$, $\mathcal{S}^{\mathcal{A}}$, $\mathcal{C}$)
    while i $\leq \mathcal{C}.\eta$ do
        while j $\leq \mathcal{S}^{\mathcal{R}}.\mathcal{N}$ do
            if $\mathcal{S}^{\mathcal{R}}.\mathcal{A}_j$ == $\mathcal{C}.\mathcal{A}_i$ then
                Append $\mathcal{S}^{\mathcal{R}}.\mathcal{A}_j$ to rAttributeSet
            Assign j $\leftarrow$ j + 1
        while k $\leq \mathcal{S}^{\mathcal{A}}.\mathcal{N}$ do
            if $\mathcal{S}^{\mathcal{A}}.\mathcal{A}_k$ == $\mathcal{C}.\mathcal{A}_i$ then
                Append $\mathcal{S}^{\mathcal{A}}.\mathcal{A}_k$ to aAttributeSet
            Assign k $\leftarrow$ k + 1
        Assign i $\leftarrow$ i + 1
    while t < $\mathcal{C}.\eta$ do
        if $\mu$(rAttributeSet[t], aAttributeSet[t]) $\prec \mathcal{C}.\mathcal{M}_t$) then
            return fail
        Assign t $\leftarrow$ t + 1
    return success
```

Fig. 2. Algorithm for parameterized service matching.

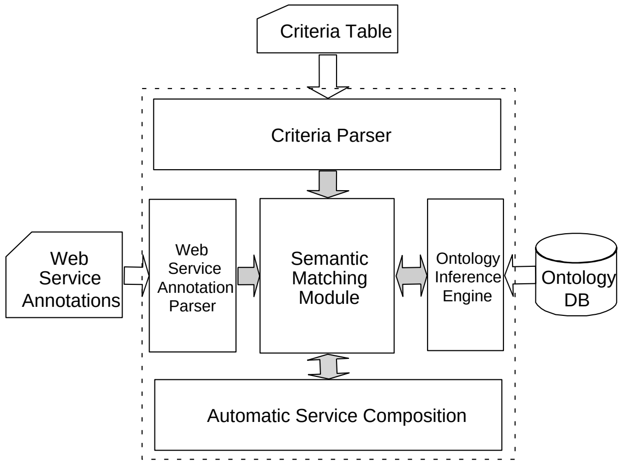

Fig. 3. Architectural diagram of PSMF

into *facts* and utilize commercial rule-based engines which use the fast Rete [6] pattern-matching algorithm. The complexity of inferring μ therefore reduces to $\mathcal{O}(|R||F||P|)$ where $|R|$ is the number of rules, $|F|$ is the number of facts, and $|P|$ is the average number of patterns in each rule. Hence the worst case complexity of our algorithm is $\mathcal{O}(\mathcal{C}.\eta \times \mathcal{S}^{\mathcal{A}}.\mathcal{N})$ + $\mathcal{O}(|R||F||P|) \asymp \mathcal{O}(|R||F||P|)$.[3]

One method for overcoming the limitations of plain capability-based matchmaking is to include attributes under service property as part of service capability. Specifically, attributes such as *Quality of Service* may be included in *Service Input* or *Service Output*. However, existing Web services representation languages such as DAML-S [7], clearly distinguish between the two, providing separate placeholders (XML tags) for each. Therefore, capability-based matchmaking precludes the use of service property as part of its match criteria.

B. Architecture

Typical activities of a semantic matching framework encompass parsing Web services interface definitions; utilizing an inference engine capable of performing, at the very least, *modus ponens*, *modus tollens*, and *resolution* inferencing techniques on ontologies to infer the similarity measure; and matching the submitted service request with the existing service advertisements. Additionally, our framework also parses the Criteria Table, and permits automatic service composition to enhance its matching ability. In Fig 3 we illustrate the architecture of our matchmaking framework.

Specification of the Criteria Table as an input parameter to the matching process facilitates an iterative approach to matching services. Specifically, at each step, a residual set of service advertisements that sufficiently match with the service request is obtained. The residual set may then serve as the input to the next step where another instance of the Criteria Table is presented to the matching process. Fig 4 gives the iterative operation of the matching engine. The **Semantic Matching Module** is transformed into a generic filter that may be called repeatedly for filtering out unsought services. Furthermore, we also provide a mechanism to post-process the match results. Either a sort using a programmer-defined comparison function, or computation of the set of non-dominated match pairs is possible.

[3]We assume the usage of large domain ontologies resulting in a large $|F|$. $|R||P|$ depends on the number of relationships that need to be inferred, and the ontology representation language.

Often, none of the service advertisements individually satisfy a particular service request. However, a composition of some services may result in a new service that closely satisfies the criteria required. The capability of this new service is usually obtained by merging the capabilities of the first and last service in the chain of composition. Its properties are usually some function of the properties of the individual component services. For instance, the *Quality of Service* of the new service is obtained by applying the MINIMUM function to the *Quality of Service* attribute of the component services.

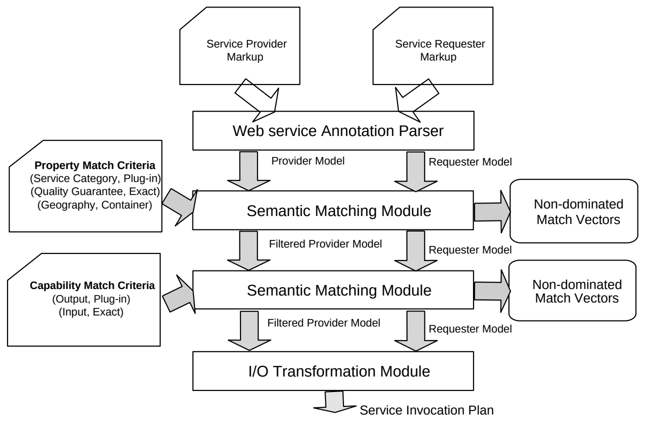

Fig. 4. The iterative operation involving successive refinement of results

Referring back to our example scenario (Definition 1), the manufacturer on realizing a deficit in its inventory may perform a *Service Category* based match to discover all suppliers that supply the desired category of parts. A single supplier may then be sought by specifying the desired *Quality of Service* and

Geographical Location. Finally, an order may be placed with the supplier by invoking its ordering Web service discovered using a service capability- based match.

C. Implementation and Performance

Our semantic matchmaking framework is implemented as a Java-based API. The popularity of both, WSDL [8] and DAML-S [7] motivated us to include parsers for both Web service representation languages. Furthermore, our matchmaker can search both, DAML-S, and semantically augmented UDDI [9] repositories. To obtain a better "hit" ratio, we attempt service compositions using a simple backward-chaining algorithm. Our matchmaking framework is designed to interface with any off-the-shelf ontological inference engine. For testing purposes we have used ABLE [10] as its pluggable rule-based inference engine.

Utilizing semantic information during the matching process significantly increases the cardinality of successful matches, as compared to strict syntactic matching. Indexing to common semantic information, such as ontologies, shields the matchmaking from the disparate terminology that may be used while creating the Web service definitions. Furthermore, the matchmaker must parse and compare a large cardinality of advertisements. The process can be significantly speeded up by indexing into specific classes of services and thus reducing the cardinality of the set of candidate advertisements. For example, we can concentrate on specific categories of services by performing a preliminary match on the *Service Category* property. A decisive capability-based match can then be performed on the filtered set of advertisements. This idea exposes the benefit of our generic matching component that may be called repeatedly with differing match criteria on a set of advertisements to progressively constrict the search-space. It also motivates the need for efficient classification of services that may be used as indexes. Such classifications, though present in UDDI repositories in the form of NAICS[4] and UNSPSC[5] are lacking in DAML-S based services.

IV. An Application - Business Process Integration and Management

A significant area of application for Web services is in inter-enterprise (B2B) and intra-enterprise (EAI) process integration and management. Several enterprises will provide a Web service based interface to their core business systems. Linking together the business processes of these enterprises translates to composing a workflow of required functionality using Web services. Fig 5 illustrates this observation.

Existing techniques for integrating business processes are manual and adhoc. These techniques must necessarily adapt to the new computing paradigm of Web services. We believe that our semantic matchmaking framework represents a candidate technology in implementing this change. Specifically, highly

[4]The North American Industry Classification System published by US Census

[5]The United Nations Standard Product and Service Classification jointly developed by UNDP/Dun & Bradstreet Corp. in 1988

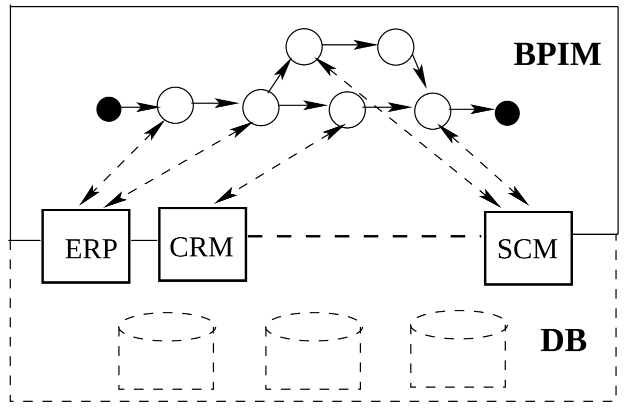

Fig. 5. BPIM Layer in Enterprise IT Infrastructure

customizable generative matchmaking will be required to implement workflows of varying functionalities. Plain capability-based matchmaking between services will prove to be inadequate for this application. The example scenario given in Definition 1 is one such process integration task that substantiates this claim.

V. Discussion

In this paper we presented a semantic matchmaking framework that utilizes a parameterized generative matchmaking algorithm. The framework serves as a key building block for choreographing Web service based workflows. Through an example scenario, we demonstrated the wider applicablity of our framework as compared to other preliminary efforts in matchmaking. Finally, we discussed a candidate industrial application of our work. Specifically, employment of semantic matchmaking will have significant contribution towards automating business process integration.

As part of future work, we intend to improve the performance of the matchmaking algorithm by utilizing better similarity measures. Additionally, we intend to develop automated workflow composition techniques by utilizing the semantic matchmaking infrastructure.

References

[1] A. Kiryakov and K. Iv, "Ontologically supported semantic matching," in *Proceedings of NoDaLiDa'99 conference. Trondheim, Norway, 1999.*, 1999. [Online]. Available: citeseer.nj.nec.com/article/kiryakov99ontologically.html

[2] M. Laukkanen and H. Helin, "Composing workflows of semantic web services," in *Workshop on Web Services and Agent-based Engineering, AAMAS*, Melbourne, Australia, 2003.

[3] K. Sycara, M. Paolucci, M. van Velsen, and J. Giampapa, "The retsina mas infrastructure," in *The RETSINA MAS Infrastructure. Technical Report CMU-RI-TR-01-05, Robotics Institute Technical Report, Carnegie Mellon, 2001.*, 2001. [Online]. Available: citeseer.nj.nec.com/sycara01retsina.html

[4] L. Li and I. Horrocks, "A software framework for matchmaking based on semantic web technology," in *Twelfth International World Wide Web Conference*, 2003.

[5] M. Paolucci, T. Kawamura, T. Payne, and K. Sycara, "Semantic matching of web services capabilities," in *International Semantic Web Conference, Sardinia, Italy, June 9-12, 2002*, June 2002.

[6] C. Forgy, "Rete: A fast algorithm for the many patterns/many objects match problem," *Artificial Intelligence*, vol. 19, no. 1, pp. 17–37, 1982.

[7] DAMLS-Coalition, “Daml-s: Web service description for the semantic web,” in *International Semantic Web Conference, Sardinia, Italy, June 9-12, 2002*, 2002. [Online]. Available: citeseer.nj.nec.com/507459.html

[8] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, “Web services description language,” Website http://www.wsdl.org, 2001.

[9] R. Akkiraju, R. Goodwin, P. Doshi, and S. Roeder, “A method for semantically enhancing the service discovery capabilities of uddi,” in *Workshop on Information Integration on the Web, IJCAI*, Acapulco, Mexico, 2003.

[10] J. P. Bigus, D. A. Schlosnagle, J. R. Pilgrim, W. N. M. Ill, and Y. Diao, “Able: A toolkit for building multiagent autonomic systems,” *IBM Systems Journal*, vol. 41, no. 3, pp. 350–371, 2002.