Matchmaking and ranking of semantic web services using integrated service profile

Natenapa Sriharee and Twittie Senivongse*

Department of Computer Engineering, Chulalongkorn University,
Phyathai Road, Pathumwan, Bangkok 10330, Thailand
E-mail: natenapa23@yahoo.com E-mail: twittie.s@chula.ac.th
*Corresponding author

Abstract: Service discovery is a key aspect in the enabling technologies for service-oriented systems, including web services. Growing attention has been paid to the content of business and service descriptions to allow services to be discovered more flexibly and accurately. This paper presents a service description model called an integrated service profile, which describes the capabilities of a service in various aspects, such as attribute-, structure-, behaviour-, and operational rule-based capabilities. An integrated service profile can be used to discover web services semantically. Criteria for considering matching between the service description and the expected capability specified in the request, with respect to each part of the profile, are proposed. A matching algorithm is based on a flexible match approach and can retrieve relevant services by using user's preference criteria. A ranking methodology with an ordinal scale is also proposed to determine the degree of matching among the matched services.

Keywords: semantic web services; matchmaking; ranking; ontology.

Reference to this paper should be made as follows: Sriharee, N. and Senivongse, T. (2006) 'Matchmaking and ranking of semantic web services using integrated service profile', *Int. J. Metadata, Semantics and Ontologies*, Vol. 1, No. 2, pp.100–118.

Biographical notes: Natenapa Sriharee is a PhD candidate at the Department of Computer Engineering, Chulalongkorn University, Bangkok, Thailand.

Twittie Senivongse is an Assistant Professor at the Department of Computer Engineering, Chulalongkorn University, Thailand.

1 Introduction

The diversity of the format and content of service descriptions within a service-oriented environment has been problematic for service consumers when looking for available services. The standard UDDI registry for web services (uddi.org, 2002) attempts to standardise business and service descriptions through a set of business and service attributes. However, the attribute set is coarse and gives only preliminary information about the service providers and the offered web services. Generally, search is by matching of name or category of business entities, business services, or tModels against the values specified in the query, such as "Find a service provider in the electronics appliance category". The search will return some information and the rest is left to the service consumer to browse the web pages of those companies to make a selection. The search does not yet support a query that is also based on semantic or behavioural information such as

> "Find an online electronics shop that sells desktop computers and is rewarded Thailand Electronics Association's Vendor award from the Ministry of Commerce. The store should accept Amex credit card and deliver the computer that I have bought to my place (in Bangkok) within 3 days."

Service description models should attract service providers to publish useful information and at the same time facilitate consumers to discover the right services. It is assumed here that service providers will do their best to please service consumers, and will advertise rich information regarding their profiles and service capabilities in order to get themselves discovered easily. This research works around the questions "What should be in a service description to allow service consumers to query more conveniently and flexibly?" and "How can such information in the service description help the consumers make a service selection?" This paper presents three important tasks that aim to answer the questions above:

- Modelling of web services descriptions:
  This task comprises a survey on service descriptions and their contents, and the result is an integrated service profile which consists of information that describe web services in terms of their attributes, semantic structure, behaviour, and operational rules. These are capability descriptions that specify various aspects of what the services can do (Oaks et al., 2003) and we use ontology to represent them. The preliminary idea of the integrated service profile has been reported in Sriharee and Senivongse (2005).

- Matchmaking of web services: An algorithm is proposed to determine whether the integrated service profiles of any service providers match to the capability expected by a service consumer. Matchmaking considers each part of the integrated service profile.
- Ranking of web services: An algorithm is proposed to determine how close a matched service is to a consumer's query. Service consumers can use the comparative ordinal scale assigned to each matched service as a suggestion for making a service selection.

Section 2 describes the conceptual model of the integrated service profile as the metadata for semantic web services. Section 3 discusses details of each part of the integrated service profile, which is represented by ontology. Section 4 considers the criteria for matchmaking of the query and the integrated service profiles, and a matchmaking example is given in Section 5. A ranking algorithm is proposed in Section 6. An evaluation of the matchmaking and ranking approaches can be found in Section 7. Section 8 presents a framework for discovery based on the integrated service profile. The paper discusses related work in Section 9 and concludes in Section 10.

2 Metadata for semantic web services: the integrated service profile

Metadata for web services give information primarily for service consumers to get to know any potential services without having to really deploy them, and therefore are useful sources of information for discovering web services. With regards to the purpose of the interaction between service providers and service consumers (Booth et al., 2004), information about attributes, characteristics, operational aspects, and deployment aspects of web services are typical web services metadata. Simple metadata for web services are modelled as attribute-based service descriptions (uddi.org, 2002; Dumas et al., 2001), meaning that the characteristics and other information about the services are described through a set of concrete attributes with corresponding attribute values. To enable more flexible and accurate discovery, semantic annotation is added to web services metadata (Martin et al., 2004; WSMO, 2004). Whether they are attribute-based or semantics-based, metadata will influence individual and organisational use of the services (Lynne and Soh, 2002), and therefore should reflect both functional and psychological needs of the individuals and organisations.

To answer the question concerning how to model the metadata for web services, we conducted an empirical survey, as reported previously in Tapabut et al. (2002), to find what information should be included in web services metadata model. Information was gathered from web services brokerage sites (such as http://www.salcentral.com, http://www.capescience.com, http://www.webserviceoftheday.com, and http://www.xmethods.com), a survey on commercial software components on the internet market (since web services can be seen as service components), and a survey on relevant research papers. Our empirical survey resulted in an attribute-based model for web services metadata; some part of it is shown in Table 1. It can be seen that some information can be easily modelled as attributes, meaning that simple attribute values can be assigned (e.g., ServiceName, Description, Award), while some refers to more complex values (e.g., interface, structure, or behaviour information). We hence see web services metadata as a combination of attribute-based information and more complex specifications on which complex analysis of the service characteristics can be conducted. As semantic annotation is a major vehicle to more flexible service discovery, this paper uses ontology as a shared formal representation (Gruber, 1993) to represent semantics of web services in those specifications.

According to the survey result, an *integrated service profile* is proposed as a metadata model for semantic web services (Figure 1). The integrated service profile comprises a number of subprofiles which maintain either the attribute-based information or the more complex capability-based information as follows:

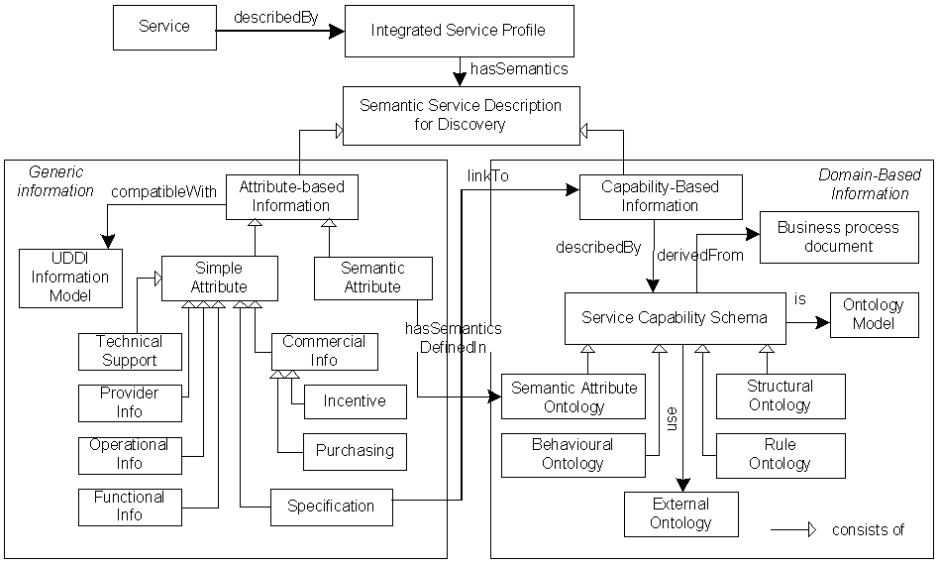
- *Attribute-based information* refers to those attributes in Table 1. This set of attributes is applicable to model web services of any application domains. It is also compatible with the attributes define in the standard UDDI information model; some attributes in the set can be mapped directly to those in the UDDI registry while some can be accommodated by an extended registry. Most of the attributes are *simple attributes* as they can be characterised by simple attribute values. Nevertheless, ontology can be useful to turn a simple attribute into a *semantic attribute* by assigning an ontological term, defined in a *semantic attribute ontology*, as its value (Sriharee et al., 2004a). For example, the value 'ThailandBestBrand' of the attribute Award may be a term in an *external ontology* (i.e., an award-related ontology), not just a simple string value. This will enable the matchmaking process to perform ontological matching, rather than string matching, when comparing the consumer's query against the service's capability. Even though semantic attributes enable more flexible matching, they are seen as a more advanced feature. Simple attributes are still maintained as part of the integrated service profile because they are a common way to describe web services; they can be applied more directly to UDDI and are more convenient for service providers to publish and for service consumers to understand. Simple attributes of a web services will be maintained by a *simple attribute profile* and semantic attributes by a *semantic attribute profile.*
- *Capability-based information* refers to the more complex aspects of web services, i.e., the capabilities, which will be represented by ontology-based *service capability schema*. The capabilities here relate to the specification-oriented attributes in Table 1 as follows:

- *Service structure* captures fundamental knowledge structure of web services of a particular domain. This is static information that service consumers would generally expect to know such as the product of the service, sales detail, and means of service delivery (Trastour et al., 2001; Li and Horrocks, 2003). Service structure is represented by a *structural ontology* in a *structural profile*.
- *Service behaviour* captures more dynamic behavioural information of web services. It is modelled as a function which may require some inputs in order to produce some outputs and effects under certain conditions (The DAML-S Services Coalition, 2002). Service behaviour is represented by a *behavioural ontology* in a *behavioural profile*.
- *Service constraint* captures constraints on service provision in terms of rules. Rules may state conditions or policies concerning the activity of web services (c.f. structural assertion rules (Hay and Healy, 2000)), and add the dynamicity to the semantics of web services. Service constraints within a particular domain are represented by a *rule ontology* in a *rule profile*, and can be associated with either the service structure or service behaviour.

Table 1 Part of the survey result on web services descriptions

Service	Operational Info		ServiceName, Version, TimeOfRelease, …
	Functional Info		Domain, Description, DevelopmentEnv, QoS, Security, …
	Commercial Info	Purchasing	Pricing, Licensing, …
		Incentive	Award, ReferenceCustomer, Promotion, Testing, …
	Technical support		Contact, FAQ, …
	Specification		Interface, Structure, Behaviour, Component, …
Provider	ProviderName, About, Domain, Certificate, …		

Figure 1 Conceptual model of the integrated service profile



As our approach uses ontology as a shared formal representation for semantics-based metadata, the ontology is modelled using a top-down approach in which the development process starts with the definition of the most general concepts followed by subsequent specialisation of the concepts (Gómez-Pérez, 1999). This paper first provides the general concepts for all capability-based metadata in terms of the upper ontologies (see Section 3). As the name implies, capability-based metadata should in fact vary according to different capabilities of web services in different domains. Experts in a particular application domain who are familiar with the nature and business processes of the domain will therefore subsequently derive, from the upper ontologies, the service structure, service behaviour, and operational constraints for the domain. Service providers in this domain can then use such shared domain ontologies as templates for publishing their own capability-based profiles. On defining ontology-based metadata, auxiliary *external ontologies* can also be imported to define some data elements which make the ontologies more complete. For example, the structural profile of an electronics appliance vendor may import an electronics appliance manufacturer ontology for the concept that represents the product model that are available at the vendor's shop.

Combining attribute-based and all capability-based profiles, the integrated service profile will be able to accommodate both conventional service discovery via attribute values matching and semantic discovery via ontological analysis on ontology-based metadata.

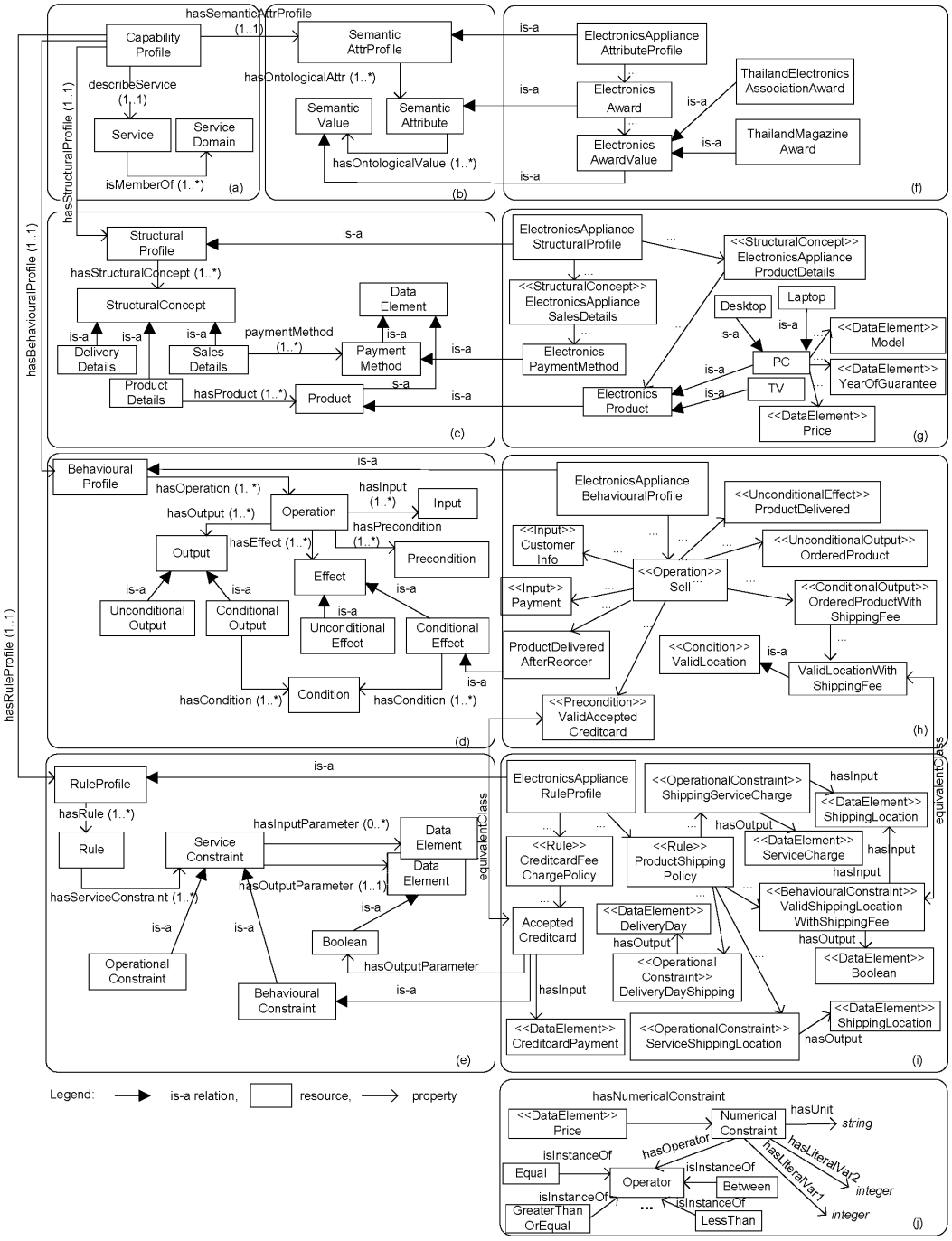
3 Service capability schema

This section focuses on modelling all capability-based profiles with ontology. As mentioned in the previous section, the conceptual model for semantic web services comprises two layers of ontology, namely the upper ontology layer and the service domain ontology layer. Service domain ontology is derived from base concepts in the upper ontology and defines new concepts that are specific to the domain. Subsequently, service providers can describe their capability-based profiles based on the domain ontologies. Figure 2 depicts the two-layer ontology model with details as follows.

- *Capability ontology* (Figure 2(a)) models a collection of web service capabilities within a domain. It is used to derive a *capability profile* which refers to the semantic attribute profile, structural profile, behavioural profile, and rule profile of a web service. The capability profile itself does not actually represent any of the service capabilities (so it will not be considered further in the matching and ranking process).
- *Upper semantic attribute ontology* (Figure 2(b)) models a number of semantic attributes whose values are ontological values. The upper ontology contains the concepts SemanticAttrProfile, SemanticAttribute, and SemanticValue and is used to derive an ontology-based *semantic attribute profile*. Figure 2(f) shows a *semantic attribute ontology* for the semantic attribute ElectronicsAward in the ElectronicsAppliance domain. The ontology defines vocabularies for the kinds of awards and the providers in the domain can use them as the values for the attribute ElectronicsAward.
- *Upper structural ontology* (Figure 2(c)) models the structure of fundamental static knowledge about a web service. It is used to derive an ontology-based *structural profile* which contains a number of StructuralConcepts including SalesDetails, ProductDetails, and DeliveryDetails. The concept ProductDetail in the upper ontology models either tangible products (e.g., a desktop PC from an electronics appliance vendor) or intangible products (e.g., information obtained from a search engine). The concepts SalesDetail and DeliveryDetail can respectively model information about payment and channels for service delivery. Figure 2(g) shows a *structural ontology* for services in the ElectronicsAppliance domain. It defines possible payment methods and vocabularies of products within the domain, with relevant product details such as model, years of guarantee, and price. Such product details are defined with the concept DataElement, meaning that they are data concept that may be imported from other external ontologies to add details to the structural ontology.
- *Upper behavioural ontology* (Figure 2(d)) models functional capability of a web service in terms of its operations. Each operation requires some inputs and produces different outputs and effects, sometimes when particular conditions are satisfied (The DAML-S Services Coalition, 2002). The upper ontology is used to derive an ontology-based *behavioural profile*. The concept Operation in the upper ontology may have some Precondition that must hold before the service can function. Outputs and effects of the Operation may be ConditionalOutput or ConditionalEffect if there are some behavioural constraints associated with them; otherwise they will be UnconditionalOutput and UnconditionalEffect. By modelling behavioural capability as a collection of operations, the behavioural profile can then be used as a semantic specification for WSDL interface specification of a web service (Christensen et al., 2001). Figure 2(h), shows a *behavioural ontology* for web services in the ElectronicsAppliance domain. It has an operation Sell which may require CustomerInfo and Payment detail as inputs. The precondition ValidAcceptedCreditcard says that the operation will function only when the customer provides a valid credit card (i.e., one of the credit cards accepted by the service). This precondition is an equivalentClass to the behavioural constraint AcceptedCreditCard in the rule ontology in Figure 2(i) (see below). The operation may return any of the unconditional or conditional outputs/effects. The conditional output OrderedProductWithShippingFee specifies that the operation may reply with the ordered product and a shipping fee which has to be paid. But this depends on whether the customer is located in a valid shipping location (i.e., the condition ValidLocationWithShippingFee); otherwise there is no fee as there will be no shipping. ValidLocationwithShippingFee is defined as an equivalentClass to the behavioural constraint ValidShippingLocationWithShippingFee in the rule ontology.
- *Upper rule ontology* (Figure 2(e)) models constraints on the provision of the service and is used to derive a *rule profile*. Each rule states a constraint or policy of the activity of the service and is modelled by the concept ServiceConstraint which may require some inputs and will be evaluated into an output value (i.e., it reads as IF (inputs are true) THEN (return output)). The concept BehaviouralConstraint refers to the constraint that requires at least one input to be evaluated and returns a Boolean output value. The concept OperationalConstraint may or may not require input and may return a non-Boolean output value. BehaviouralConstraint is aimed for describing preconditions and conditions associated to outputs and

effects in the behavioural profile (Sriharee and Senivongse, 2003). We assume that service consumers will be guided by the discovery framework to supply inputs needed for evaluation of any service constraints within the domain of interest. Figure 2(i) shows a *rule ontology* for the ElectronicsAppliance domain with the policies on product shipping and on credit cards used for shipping payment. For ProductShippingPolicy, the service may have operational constraints on ServiceShippingLocation (which returns the locations covered in the shipping area), on DeliveryDayShipping (which returns the number of days required for shipping), and on ShippingServiceCharge (which returns the shipping charge based on the shipping location and value of payment). The behavioural constraint ValidShippingLocationWithShippingFee may return true or false depending on whether the customer is located in the area of shipping. For CreditCardFeeChargePolicy, the behavioural constraint AcceptedCreditCard may return true or false depending on whether the customer presents a credit card that is one of those accepted.

Figure 2 Ontology model for service metadata: (a) Capability ontology; (b)–(e) Upper ontologies for semantic attribute, structural, behavioural, and rule profiles respectively; (f)–(i) Domain ontologies for electronics appliance domain: semantic attribute, structural, behavioural, and rule ontologies respectively and (j) Ontology for numerical constraints



These five kinds of ontologies for a particular domain will be defined by domain experts. We assume all service providers in the same domain share the same capability-based ontologies and do not consider the case that ontologies may change. Service providers will publish their profiles according to these five ontologies. We can use OWL (W3C, 2004) as an ontology language since several tools exist and it is recommended by W3C. Note that our approach also accommodates numerical constraints on the concepts defined in the structural or rule ontologies. For example, an electronics appliance vendor named PowerBuy may want to publish in the structural profile that the price of its PCs is between 20,000–80,000 bahts (e.g., $20{,}000 \leq \text{Price} \leq 80{,}000$ bahts), or, in the rule profile, that the delivery day is no more than three days (e.g., DeliveryDay ≤ 3 days). Since OWL does not yet provide for this kind of constraint expressions (Pan and Horrocks, 2004), Figure 2(j) gives additional ontology for representing a simple formal expression for such numerical constraints (Sriharee et al., 2004b).

4 Matching criteria for the integrated service profile

This section explains matching criteria for determining whether an integrated service profile of a provider matches the query of a consumer. Matching is based on the comparison between two relation expressions, one in a particular profile of the provider and the other in the query. The provider's profiles and the query can be seen as a collection of these relation expressions. Each relation expression is in the form of <subject, property, object> where subject refers to either the query or one of the provider's profiles (and will be omitted for brevity), property is the service characteristic to be compared, and object is the value of the property. For most profiles which are ontology-based (except for the simple attribute profile), this form corresponds to an RDF expression.

The following Sections 4.1–4.6 explain matching criteria which consider all aspects of the integrated service profile. Results from these will lead to a classification of matching types and assignment of their ordinal scale in Section 4.7. It is assumed that any services that do not publish any aspects requested in the query will not be considered in the matching process.

4.1 Matching ontological concepts

As most profiles (except for the simple attribute profile) are ontology-based, matching by subsumption and equivalence is the basis for matching ontological concepts in the query and the provider's profile (Baader et al., 2003). This approach has been adopted in Sycara et al. (2002), Paolucci et al. (2002), Trastour et al. (2002), Li and Horrocks (2003) and Di Noia et al. (2003). In Resnik (1995) and Andreasen et al. (2003), a weaker match of ontological concepts, called partial match, is defined for two concepts that have a shared node in IS-A taxonomy and do not have a subsumption relationship between them. The degree of matching is determined between two concepts as described below.

For two relation expressions of the same property, one in the query and the other in the provider's profile, let C_Q be the property value specified in the query and C_P be the one in the profile:

- *If* $C_Q \equiv C_P$ *then* C_P *is an exact match for* C_Q, where ≡ means is equivalent to. For example from Figure 2(g), the provider who sells Desktop will be an exact match for the query that also requests for Desktop.
- *If* $C_P \sqsubseteq C_Q$ *then* C_P *is a specialised match for* C_Q, where $\sqsubseteq$ means is subsumed by (i.e., C_P is more specific than C_Q). In this case, the query may specify a generic concept while the profile defines a specific concept. For example from Figure 2(g), the profile that sells either Notebook or Desktop will be a specialised match for the query that requests for PC.
- *If* $C_Q \sqsubseteq C_P$ *then* C_P *is a generalised match for* C_Q. This means the concept in the query is more specific than, and is subsumed by, the one in the profile. For example from Figure 2(g), the profile that sells PC will be a generalised match for the query that requests for a Desktop.
- *If* $(C_Q \not\sqsubseteq C_P) \wedge (C_P \not\sqsubseteq C_Q) \wedge (C_Q \sqsubseteq C_C) \wedge (C_P \sqsubseteq C_C)$ *then* C_P *is a partial match for* C_Q, where $\not\sqsubseteq$ means is not subsumed by and C_C is a node in the same IS-A taxonomy. This means it is acceptable for the concept in the profile to be a match for the concept in the query provided that the two concepts have common characteristics through a common parent concept. For example from Figure 2(g), the profile that sells Laptop will be partial match for the query that requests for Desktop.
- *If none of the above relationships exist then* C_P *is a failed match for* C_Q.

4.2 Matching numerical constraints

As mentioned earlier, service providers and consumers may put numerical constraints on relation expressions in the context of the structural or rule ontology. For example, the relation expression on the price of the product PC may be published or queried with such a constraint that the price is between 30,000–50,000 bahts. Or the relation expression on the rule for the number of delivery day for shipping may be published or queried with a constraint that it is less than three days. Matching two numerical constraints compares the intervals of the possible values that are defined in the constraints. The degree of matching for numerical constraints can be determined as described below.

For two relation expressions of the same property, let N_Q be a nonempty set of numerical constraint values of the relation expression in the query (R_Q), and N_P be a nonempty

set of numerical constraint values of the relation expression in the profile (R_P):

- *if* $N_P \subseteq N_Q$ *then* R_P *is an exact match for* R_Q
- *if* $N_Q \subseteq N_P$ *then* R_P *is a plug-in match for* R_Q
- *if* $(N_P \cap N_Q \neq \phi) \wedge (N_P \subsetneq N_Q) \wedge (N_Q \subsetneq N_P)$ then R_P *is a weak match for* R_Q
- *if* $N_P \cap N_Q = \phi$ *then* R_P *is a failed match for* R_Q.

4.3 Matching sets of ontological values

Service providers or consumers may publish or query multiple ontological values on any relation expressions related to the semantic attribute and structural ontologies. For example, the relation expression on the award that the service has obtained may be published or queried with both values ThailandElectronicsAssociationAward and ThailandMagazineAward. Or the relation expression on the product for sale by the service may be both Desktop and TV. Matching two sets of ontological values comes down to matching each of the values in the two sets based on ontological matching (Constantinescu and Faltings, 2003).

For two relation expressions of the same property, let D_Q be a nonempty set of ontological values of the relation expression in the query (R_Q), and D_P be a nonempty set of ontological values of the relation expression in the profile (R_P):

Definition: The profile will satisfy a set of ontological values match on the query if there exists an ontological match (Section 4.1) between each concept in the query and a concept in the profile. This is denoted by

$$SetOfOntoValsMatch(R_Q, R_P) = true \Leftrightarrow$$
$$\forall i, \exists j : (i \in D_Q) \wedge (j \in D_P) \wedge (i \otimes j)$$

where $\otimes$ means having a kind of the ontological match in Section 4.1 (i.e., *exact*, *specialised*, *generalised*, *partial*).

4.4 Matching service constraints

Service providers or consumers may publish or query on the values of service constraints in the rule ontology. There are two cases for considering matching between two sets of constraint values: matching operational constraints and matching behavioural constraints. For example, the operational constraint ServiceShippingLocation of a service may be published to return the concepts Bangkok, Chiang Mai, and Phuket on evaluation. If the query is for the service that provides shipping to Bangkok and Chiang Mai, the service would match. The case of behavioural constraints is more complex as they are conditions that are associated with the behavioural profile (i.e., precondition, conditional output, conditional effect). So matching of behavioural constraints will be used for determining matching of precondition, conditional output, and conditional effect. Although behavioural constraints require input parameters to evaluate to either true or false, the behavioural profile concerns only when they evaluate to true, by which the precondition will hold and the conditional output and conditional effect will result. For example, in Figure 2(h), the conditional output OrderedProductWith ShippingFee will result only if the condition ValidLocation WithShippingFee is true. Since ValidLocationWithShipping Fee is an equivalentClass to the behavioural constraint ValidShippingLocationWithShippingFee, we first evaluate this behavioural constraint by using the location (specified in the query) as the input parameter (say, Bangkok and Chiang Mai). If the location is among the valid values, defined by the service, for this constraint (e.g., Bangkok, Chiang Mai, Phuket), it will evaluate to true which means the equivalent ValidLocationWithShippingFee is also true, and the result is the output OrderedProductWithShippingFee will be produced. In other words, the output OrderedProduct WithShippingFee of this service satisfies the query based on the location input from the query. For behavioural constraints, matching is therefore considered against the values of the input parameter of the constraint.

Since the values related to the evaluation of a service constraint may in fact be either a range of numerical values or a set of ontological concepts, we can adopt the matching rules in Sections 4.2 and 4.3 here. For two relation expressions of the same property, let O_Q be a non-empty set of ontological outputs or of ranges of numerical output for an operational constraint specified in the query (R_Q), O_P be a non-empty set of ontological outputs or of ranges of numerical output for an operational constraint in the profile (R_P), I_Q be a non-empty set of ontological inputs or of ranges of numerical input for a behavioural constraint specified in the query (R_Q) where the constraint is evaluated to true, and I_P be a non-empty set of ontological inputs or of ranges of numerical input for a behavioural constraint specified in the profile (R_P) where the constraint is evaluated to true:

Definition: The profile will satisfy a set of constraints match on the query if, depending on whether the constraints are operational or behavioural, the output or input for each of the constraint evaluation of the query matches one in the profile. This is determined by

i $SetOfOperationalConstrsMatch(R_Q, R_P) = true \Leftrightarrow$
$\forall i, \exists j : (i \in O_Q) \wedge (j \in O_P) \wedge (i \odot j)$

ii $SetOfBehaviouralConstrsMatch(R_Q, R_P) = true \Leftrightarrow$
$\forall i, \exists j : (i \in I_Q) \wedge (j \in I_P) \wedge (i \odot j)$

where $\odot$ means either having a kind of the ontological match in Section 4.1 (i.e., *exact*, *specialised*, *generalised*, *partial*) or having a kind of the numerical constraint match in Section 4.2 (i.e., *exact*, *plug-in*, *weak*).

4.5 Matching behavioural profiles

Matching behavioural profiles determines whether the behavioural capability of the service can satisfy or realise

the behavioural requirement in the query (Liskov and Wing, 1994; Zaremski and Wing, 1997; Wickler, 1999). Intuitively, the service will satisfy the query if, given the precondition and input from the query, the service can accept and operate successfully, giving out satisfied output or effect to the query. Ontological matching in Section 4.1 is used to determine matching for each relation expression in the behavioural profile as follows:

- *Operation, Precondition, Output, and Effect match.* An ontological concept signifying either an operation, precondition, output, or effect in the profile will match to its counterpart in the query if they have a kind of ontological match in Section 4.1 (i.e., *exact, specialised, generalised, partial*).
- *Input match*. Ontological match in Section 4.1 is also used to determine matching between one input concept in the profile and another in the query. It is interesting to note that, unlike other aspects, *generalised* match is a better match than *specialised* match in the case of input. This means the service's operation can perfectly operate with the query's input which is more specific than what it expects. This is compared to the case when the service's operation expects a more specific input than what supplied by the query.

Let $\mathbb{R}_Q$ and $\mathbb{R}_P$ be sets of behavioural relation expressions which comprise an operation, a set of inputs, a set of outputs, a set of preconditions, and a set of effects.

Definition: The profile will satisfy a behavioural match on the query if all the behaviour expected by the query is among the behaviour that the profile exhibits. This is determined by

$$BehaviouralMatch(\mathbb{R}_Q, \mathbb{R}_P) = true \Leftrightarrow$$
$$(\mathbb{R}_Q \subseteq \mathbb{R}_P) \wedge (\forall i, \exists j : (i \in \mathbb{R}_Q) \wedge (j \in \mathbb{R}_P) \wedge (i \otimes j))$$

where $\otimes$ means having a kind of the ontological match in Section 4.1 (i.e., *exact*, *specialised*, *generalised*, *partial*).

Note that the service may publish more behavioural information than the query. For example, the service may require more number of inputs than those in the query. We do not consider this number issue in the matching process; as long as the service has the inputs that can match to the query's inputs, the service satisfies the query in that respect. Suppose that finally the consumer selects to use this service, the consumer can study from the WSDL of the service to find out what more inputs are needed.

Since a behavioural profile contains relation expressions that relate to many aspects of the behavioural model, we then have to determine matching for all of associated semantic elements. In the case that semantic elements are preconditions, outputs, or effects with associated behavioural constraints in the rule ontology, they will match only if they can also satisfy behavioural constraint match as mentioned in Section 4.4.

4.6 Matching simple attributes

Matching of simple attributes in the simple attribute profile is based on comparing descriptive string values of the attributes and determining their similarity. We adopt an approximate string matching technique called *q*-grams (Ukkonen, 1992; Gravano et al., 2001; Navarro, 2001). The basic idea of *q*-grams is 'sliding' a window of length *q* over the characters of string σ. To achieve a better comparison, words with no information value are removed from the descriptive string before processing. It is also possible to provide a list of keywords for a particular domain to help specifying attribute values when publishing or querying. Matching descriptive attribute values can be implemented by extracting terms, which are likely to match the listed keywords, from the profile and the query. Extraction can be implemented by substring match, and later use *q*-grams for computing similarity.

Let *q* be length of *q*-grams, σ be a set of *n* keyword terms extracted from the value of a simple attribute in the query, a set G_{σ_i} be *q*-grams of a string σ_i where $\sigma_i \in \sigma$, Ω be a set of *m* keyword terms extracted from the value of the same simple attribute in the profile, and a set G_{Ω_j} be *q*-grams of string Ω_j where $\Omega_j \in \Omega$. The similarity score between σ and Ω is computed by

$$SimSimpleAttribute(\sigma, \Omega) = \left| (\bigcup_{i=1}^{n} G_{\sigma_i}) \cap (\bigcup_{j=1}^{m} G_{\Omega_j}) \right| \Big/ \left| (\bigcup_{i=1}^{n} G_{\sigma_i}) \right|.$$

For example, given σ = {electronics, retail}, Ω = {retail}, and length *q* is 3, *q*-grams of the string 'electronics' is {##e,#el,ele,lec,ect,ctr,tro,ron,oni,nic,ics,cs#,s##}, and *q*-grams of the string 'retail' is {##r,#re,ret,eta,tai, ail,il#,l##}. Therefore the similarity score between σ and Ω is 0.38. The similarity score helps classify the types of simple attribute matching such as: *StrongMatch* [0.75, 1], *OptimisticMatch* [0.50 – 0.75), *RelaxedMatch* [0.25, 0.50), and *Failed* [0, 0.25).

4.7 Ordinal scale of profile matching

Table 2 Types of matching and match scores

Match type	*Match score*
Ontological match (Section 4.1)	exact = 4, specialised = 3, generalised = 2, partial = 1, failed = 0
Input match of behavioural profile (Section 4.5)	exact = 4, generalised = 3, specialised = 2, partial = 1, failed = 0
Numerical constraint match (Sections 4.2 and 4.4 (for numerical values))	exact = 3, plug-in = 2, weak = 1, failed = 0
Set of values match (Sections 4.3 and 4.4 (for ontological values))	satisfied = 1, failed = 0
Simple attribute match (Section 4.6)	strong = 3, optimistic = 2, relaxed = 1, failed = 0

Table 2 summarises the types of matching and ordinal scale which represents match scores, from the strongest to the weakest match.

5 Example of matchmaking

Suppose there is a query for a web service of a retail electronics shop which sells desktop computers with the price range between 15,000 – 30,000 bahts. The shop must receive Thailand Electronics Association Award and accept credit card payment. The consumer also needs the desktop to be delivered to Bangkok within 3 days. Fee charge for delivery is acceptable but should be less than 200 bahts. We present the query ($\mathbb{Q}$) as above with the relation expressions below. Note that each expression is subscripted by a profile symbol; α, γ, ρ, and β represents simple or semantic attribute profile, structural profile, rule profile, and behavioural profile respectively. The superscript denotes the context of the relation expression; *S* refers to a simple attribute, *C* refers to a constraint which may be either a numerical, behavioural, or operational constraint, and ϕ refers to a single concept. For the behavioural profile, the superscripts Δ, *I*, *O*, *P*, *E* respectively refer to operation, input, output, precondition, and effect:

$\mathbb{Q}$ = {hasDescription('electronics, retail')$^{S}_{\alpha}$,

hasAward(ThailandElectronicsAssociationAward)$^{\phi}_{\alpha}$,

hasProduct(Desktop)$^{\phi}_{\gamma}$,

hasPrice(Desktop, *Between*, 15000, 30000, baht)$^{C}_{\gamma}$,

hasServiceConstraint(DeliveryDayShipping, Bangkok, DeliveryDay, *LessThanOrEqual*, 3, day)$^{C}_{\rho}$,

hasServiceConstraint (ShippingServiceCharge, Bangkok, ServiceCharge, *LessThanOrEqual*, 200, baht)$^{C}_{\rho}$,

hasOperation (SellElectronicsProduct)$^{\Delta}_{\beta}$,

hasPrecondition(ValidAcceptedCreditcard)$^{P}_{\beta}$,

hasInput(CreditcardPayment)$^{I}_{\beta}$,

hasOutput(OrderedProduct)$^{O}_{\beta}$,

hasEffect(ProductDelivered)$^{E}_{\beta}$ }

The integrated service profiles of two candidate services S_1 and S_2 are depicted in Figure 3. These are instance profiles, so ontological matching is considered from the base concept of each individual resource. Some IS-A hierarchies that represent knowledge in the profile ontologies of the ElectronicsAppliance domain are shown in Figure 4. Note that it is possible that a single concept in the query may match to multiple concepts in the profile. For example, the query that asks for a product Desktop could match to both products Desktop and Laptop which are published in the profile, but with a different strength (i.e., exact vs. partial match). We consider the strongest match in this case.

Figure 3 Integrated service profiles of two candidate services

S_1 = (hasDescription("*In 2003, PowerBuy is the first retail electronics store in Thailand—our goal was to provide the community with a new technology that would change the face of consumer electronics forever.*")

hasAward.ThailandElectronicsAssociationAward) α

(hasProduct.{Desktop, Laptop, TV, …}

Desktop.hasPrice($\geq$,15000, baht)

Laptop.hasPrice($\geq$,20000, baht)

TV.hasPrice($\geq$,3000, baht)) γ

(hasServiceConstraint.ShippingServiceCharge
hasInputParameter.ShippingLocation
hasOutputParameter.ServiceCharge
Constraint1: If ShippingLocation=Bangkok Then ServiceCharge=100 baht
Constraint2: If ShippingLocation=Chiangmai Then ServiceCharge=500 baht
hasServiceConstraint.DeliveryDayShipping
hasInputParameter.ShippingLocation
hasOutputParameter.DeliveryDay
Constraint1: If ShippingLocation=Bangkok Then DeliveryDay<=2 day

Constraint2: ….) ρ

(hasOperation.SellElectronicsProduct
hasPrecondition.ValidAcceptedCreditcard
hasInput.CreditcardPayment
hasOutput.OrderedProductWithoutShippingFee
hasCondition.ValidLocationWithoutShippingFee
hasOutput.OrderedProductWithShippingFee
hasCondition.ValidLocationWithShippingFee

hasEffect.ProductDelivered) β

S_2 = (hasDescription("*Comworld Co., Inc is Thailand's number-one specialty retailer of personal computers and entertainment software and appliances.*")

hasAward.ThailandMagazineAward) α

(hasProduct.{Desktop, Laptop, …}

Desktop.hasPrice($\geq$,10000, baht)

Laptop.hasPrice($\geq$,29900, baht)) γ

(hasServiceConstraint.ShippingServiceCharge
hasInputParameter.OrderedValue
hasOutputParameter.ServiceCharge
Constraint1: If ShippingLocation=Bangkok Then ServiceCharge=150 baht
Constraint2: If ShippingLocation=PhuketThen ServiceCharge=500 baht

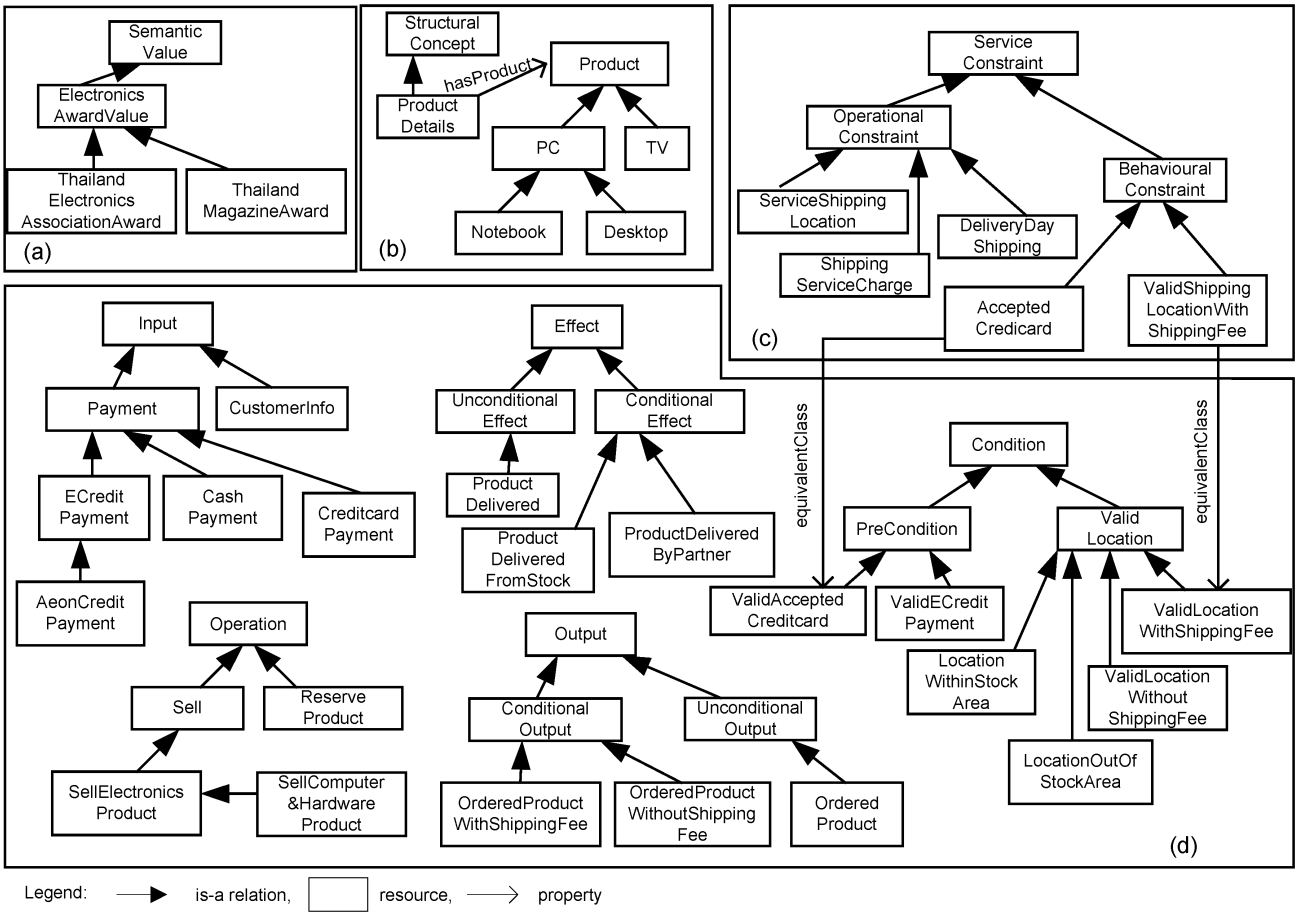
hasServiceConstraint.DeliveryDayShipping
hasInputParameter.ShippingLocation
hasOutputParameter.DeliveryDay
Constraint1: If ShippingLocation=Bangkok Then DeliveryDay<=5 day

Constraint2: ….) ρ

(hasOperation.SellComputer&HardwareProduct.
hasPrecondition.ValidECreditcardPayment
hasInput.ECreditPayment
hasOutput.OrderedProductWithoutShippingFee
hasCondition.ValidLocationWithoutShippingFee
hasOutput.OrderedProductWithShippingFee
hasCondition.ValidLocationWithShippingFee
hasEffect.ProductDeliveredFromStock
hasCondition.LocationWithinStockArea
hasEffect.ProductDeliveredByPartner

hasCondition.LocationOutOfStockArea) β

Figure 4 Fragment of ontologies for the profiles



By comparing $\mathbb{Q}$ against S_1 and S_2 and assuming that any evaluation required to evaluate the behavioural constraints of the two services are valid, matching results between $\mathbb{Q}$ and S_1, and $\mathbb{Q}$ and S_2 are reported as follows:

$$Match(\mathbb{Q}, S_1) = \{strong_{hasDescription}, exact_{hasAward}, exact_{hasProduct}, plug\text{-}in_{hasPrice}, exact_{hasDeliveryDayShipping}, exact_{hasServiceCharge}, exact_{hasOperation}, exact_{hasPrecondition}, exact_{hasInput}, partial_{hasOutput}, exact_{hasEffect}\}$$

$$Match(\mathbb{Q}, S_2) = \{relaxed_{hasDescription}, partial_{hasAward}, exact_{hasProduct}, plug\text{-}in_{hasPrice}, plug\text{-}in_{hasDeliveryDayShipping}, exact_{hasServiceCharge}, specialised_{hasOperation}, partial_{hasPrecondition}, partial_{hasInput}, partial_{hasOutput}, partial_{hasEffect}\}$$

6 Ranking methodology

After the matchmaking process discovers all the web services whose characteristics and capabilities match to what expected by the query, the ordinal scale of match types in Section 4.7 is used by the ranking process to rank all those matched services based on user preference criteria (e.g., Larichev, 2001). Service consumers can specify any of the following preference criteria for matching and ranking.

- *Match preference*. This criterion can be set to define a preference when considering matching on a particular relation expression. The preference is specified in terms of the weakest acceptable match type. For example, the service consumer may query for the product PC and set a match preference to *specialised*. So the services that publish the product with the same concept, i.e., PC (by *exact* match), and more specific concepts, i.e., Desktop and Laptop (by *specialised* match), will match to the query.
- *Feature priority preference*. This criterion can be set to define a significance that one relation expression has over the others within the same profile. This preference setting can help overcome a problem of conflicting ordinal scale of match types among several relation expressions. For example, the query specifies two relation expressions on Award and Description in the context of attributes. Suppose two candidate services have ordinal scale match as (*specialised*, *strong*) and (*exact*, *optimistic*) respectively, this can be problematic for ranking. By specifying a feature preference such that the consumer gives priority to Award over Description, the second candidate service will be ranked higher than the first one.

- *Profile priority preference*. This criterion can be set to define a significance that one profile of the service has over the others. It can be used in a similar way to the feature priority preference but is for problematic ranking across profiles. For example, the query specifies a relation expression on Award in the semantic attribute profile and another on Product in the structural profile. Suppose two candidate services have ordinal scale match as (*specialised*, *exact*) and (*exact*, *specialised*), this can be problematic for ranking. By specifying a profile priority preference such that the consumer gives priority to the semantic attribute profile over the structural profile (denoted by $\gamma \prec \alpha$), the second candidate service will be ranked higher than the first one.

Similarly to resolving ordinal scale conflict, feature priority preference and profile priority preference can be used to refine ranking. When two services have the same ranking order and a priority preference is set, the priority can be used to break the tie by further determining *sub-ranking*. In other words, sub-ranking is only for a more refined ordering within the same rank order with a similar match score, and it will not be considered if the priority preference is not set. In such a case, the services will be assumed to be ranked as equal; further ranking consideration, should the need arises, is left to the consumer.

For simplicity, only some parts from the example in Section 5 are taken to show how ranking is applied. Assume that the query now consists of the following six relation expressions: hasDescription, hasProduct, hasServiceConstraint(DeliveryDayShipping), hasOperation, hasOutput, and hasEffect. The service consumer sets a match preference on each relation expression as ($relaxed_{hasDescription}$, $exact_{hasProduct}$, $plug\text{-}in_{hasServiceConstraint(DeliveryDayShipping)}$, $specialised_{hasOperation}$, $partial_{hasOutput}$, $partial_{hasEffect}$). The feature priority preference is set for the behavioural profile and specifies the priority, from the least to most, as (effect, output, operation), which is denoted by $E \prec O \prec \Delta$. The profile priority preference is set to give equal priority to the structural, rule, and behavioural profiles, and these three has a priority over the attribute profiles (denoted by $\alpha \prec (\gamma \approx \rho \approx \beta)$).

Ranking methodology performed consists of the following steps:

i For each relation expression in the query, determine possible match scores for each of them with regards to their match preference. For six relation expressions of the query $\mathbb{Q}$ that we now focus, the relation expression hasDescription requires simple attribute matching with a match preference set to *relaxed*. Therefore its possible matches from the strongest to weakest according to Section 4.7 are {*strong*, *optimistic*, *relaxed*}. This corresponds to the $MatchScores_{hasDescription} = \{3, 2, 1\}$. Possible match scores for other relation expressions will be determined in a similar manner. Hence, $MatchScores_{hasProduct} = \{4\}$, $MatchScores_{hasServiceConstraint(DeliveryDayShipping)} = \{3, 2\}$, $MatchScores_{hasOperation} = \{4, 3\}$, $MatchScores_{hasOutput} = \{4, 3, 2, 1\}$, and $MatchScores_{hasEffect} = \{4, 3, 2, 1\}$.

ii For each profile with several relation expressions related to it, consider as follows:

- Define all possible match patterns for the profile. A match pattern is an n-tuple of the match scores from all related relation expressions which is denoted by

  $MatchPattern = (ms_1, \ldots, ms_n)$

  where n = the number of related relation expressions

  ms_i = *a match score value taken from* $MatchScores_i$, $i = 1, \ldots, n$.

  The *MatchPatterns* for the behavioural profile in the example will be (4,4,4), (4,4,3), (4,4,2), (4,4,1), (4,3,4), (4,3,3), (4,3,2), (4,3,1), (4,2,4) etc. *MatchPattern* = (4,4,4) says that there might be a web service with a behavioural profile that matches to the query with a match score 4 on operation, match score 4 on output, and match score 4 on effect.

- Classify all possible match patterns to their rank order. This is determined by summation of the match scores in each *MatchPattern*, denoted by *MatchPatternScore(MatchPattern)*. For example, *MatchPatternScore*((4,4,4)) = 12, *MatchPatternScore*((4,4,3)) = 11, and *MatchPatternScore*((4,4,2)) = 10. Different values of *MatchPatternScore* will determine the rank orders, and different *MatchPattern*s which have the same *MatchPatternScore* will falls into the same rank order. We can compute the number of possible rank orders by

  $$NumberofRankOrders = Max(MatchPatternScore_1, \ldots, MatchPatternScore_n) - Min(MatchPatternScore_1, \ldots, MatchPatternScore_n) + 1$$

  where n = the number of possible match patterns.

  In the example, the maximum *MatchPatternScore* is 12 (from *MatchPattern* = (4,4,4)) and the minimum is 5 (from *MatchPattern* = (3,1,1)). So the number of rank orders in the behavioural profile is 8. Figure 5(a) shows only the top three rank orders; the highest *MatchPatternScore* = 12 will be the top rank order 1, followed by the lower scores with lower rank orders. Each rank order has a number of match patterns assigned to it. From this assignment, we can see that a web service whose behavioural profile has *MatchPattern* = (4,4,4) would be ranked higher than the one with *MatchPattern* = (4,4,3).

- Refine ranking by determining sub-ranking based on the specified feature priority preference. If the service consumer specifies feature priority preference, it can help determine relative ranking between match patterns within the same rank order. In Figure 5(a), pattern number 2 (i.e., *MatchPattern* = (4,4,3)) is under the same rank order as pattern number 3 (i.e., *MatchPattern* = (4,3,4)) so primarily they are ranked equal. But with the feature priority preference $E \prec O \prec \Delta$ set for the behavioural profile, sub-ranking can be performed. Figure 5(a) also shows an example of a sub-ranking table for rank order 2 (*MatchPatternScore* = 11) and rank order 3 (*MatchPatternScore* = 10). In the sub-ranking table for rank order 2, pattern number 2 wins over pattern number 3 (because it has a higher match score for output), pattern number 2 wins over pattern number 4 (because it has a higher score match for operation), and pattern number 3 wins over pattern number 4 (because it has a higher score match for operation).

  Apart from the behavioural profile β, the query $\mathbb{Q}$ in our example above also involves the simple attribute profile α, structural profile γ, and rule profile ρ. We have to determine match patterns, rank orders, and sub-ranking tables for these profiles as well. In summary,

  For α, rank order 1 (*MatchPatternScore* = 3): *MatchPattern* = (3)
  rank order 2 (*MatchPatternScore* = 2): *MatchPattern* = (2)
  rank order 3 (*MatchPatternScore* = 1): *MatchPattern* = (1)

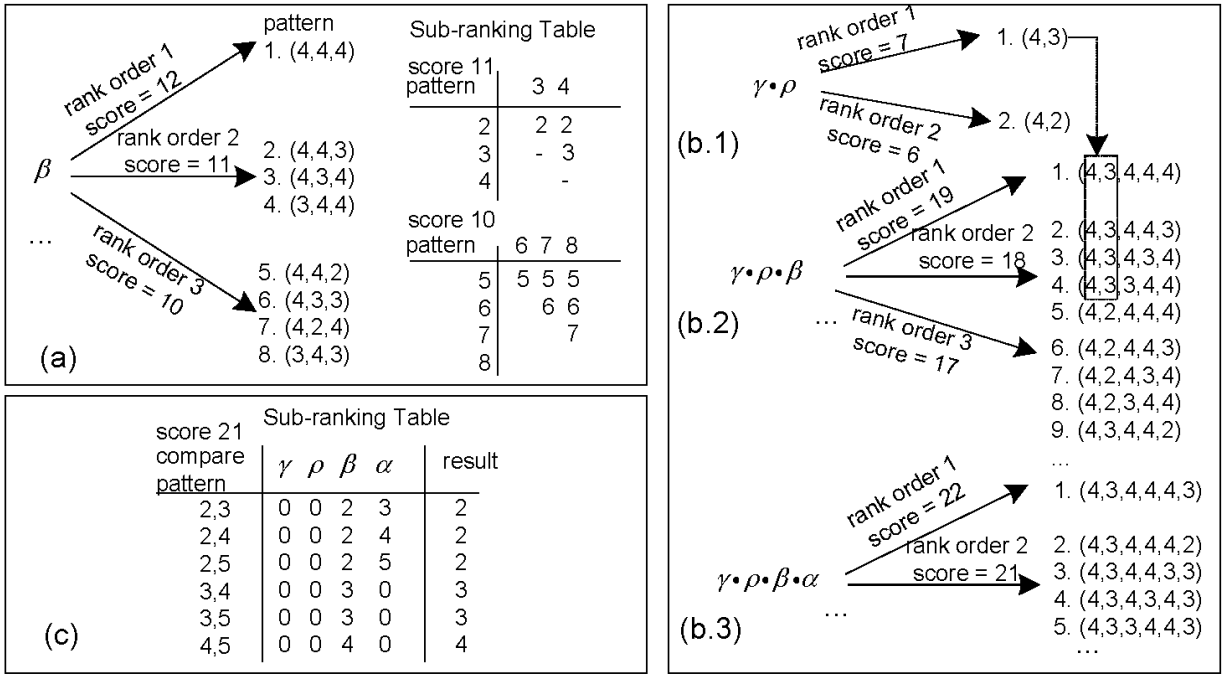
  For γ, rank order 1 (*MatchPatternScore* = 4): *MatchPattern* = (4)

  For ρ, rank order 1 (*MatchPatternScore* = 3): *MatchPattern* = (3)
  rank order 2 (*MatchPatternScore* = 2): *MatchPattern* = (2)

  Remark: For each of these three profiles, each of its rank orders has only one *MatchPattern*, therefore sub-ranking tables are not necessary.

iii Combine different profiles and determine match patterns, rank orders, and sub-ranking. This step is similar to step (ii) but is done across the profiles and the process is incremental. In Figure 5(b.1), we start with combining the structural profile γ and rule profile ρ first. All possible match patterns are defined based on all *MatchPattern* under these two profiles (which have been generated in step (ii)). Therefore, we obtain *MatchPattern* = (4,3) and *MatchPattern* = (4,2) for the combination $\gamma \cdot \rho$ with *MatchPatternScore* = 7 and *MatchPatternScore* = 6 respectively. The match patterns from $\gamma \cdot \rho$ will be used to define match patterns when the behavioural profile is added to the combination. Figure 5(b.2) shows some match patterns and rank orders for the combination $\gamma \cdot \rho \cdot \beta$. And subsequently, the simple attribute profile is combined into $\gamma \cdot \rho \cdot \beta \cdot \alpha$ in Figure 5(b.3).

Figure 5 Example of ranking



After the profiles are combined, we can similarly determine relative sub-ranking for match patterns within the same rank order. According to the profile priority preference $\alpha \prec (\gamma \approx \rho \approx \beta)$ set by the example, the sub-ranking table for rank order 2 (*MatchPatternScore* = 21) in the final combination $\gamma \cdot \rho \cdot \beta \cdot \alpha$ can be created; part of it is shown in Figure 5(c). Each row of the table compares two match patterns and determines which one wins over the other with regards to each profile. In the case that there are conflicts, the profile priority preference is considered. Considering pattern number 2 (i.e., *MatchPattern* = (4,3,4,4,4,2)) and pattern number 3 (i.e., *MatchPattern* = (4,3,4,4,3,3)), the two are equal (i.e., no one wins) with regards to the profiles γ and ρ. However, pattern number 2 wins over pattern number 3 regarding to β, while pattern number 3 wins under α. This conflict is resolved by the profile priority preference; the sub-ranking table shows the overall result such that pattern number 2 wins over pattern number 3 because β has higher priority than α. Pattern number 2 hence will be ranked higher than pattern number 3 in the sub-ranking.

Looking back at the matching results in Section 5 and the shortened query $\mathbb{Q}$ with the relation expressions: {hasDescription $^{S}_{\alpha}$, hasProduct $^{\phi}_{\gamma}$, hasServiceConstraint (DeliveryDayShipping) $^{C}_{\rho}$, hasOperation $^{\Delta}_{\beta}$, hasOutput $^{O}_{\beta}$, hasEffect $^{E}_{\beta}$}, we obtain the match results for the service S_1 and S_2 as follows.

$Match(\mathbb{Q}, S_1) = \{strong_{hasDescription}, exact_{hasProduct}, exact_{hasDeliveryDayShipping}, exact_{hasOperation}, partial_{hasOutput}, exact_{hasEffect}\}$

$Match(\mathbb{Q}, S_2) = \{relaxed_{hasDescription}, exact_{hasProduct}, plug\text{-}in_{hasDeliveryDayShipping}, specialised_{hasOperation}, partial_{hasOutput}, partial_{hasEffect}\}$

Match score for S_1 is $(3 + 4 + 3 + 4 + 1 + 4) = 19$ and for S_2 is $(1 + 4 + 2 + 3 + 1 + 1) = 12$. So S_1 is in a higher rank and closer to $\mathbb{Q}$, than S_2.

7 Matchmaking and ranking evaluation

7.1 Matchmaking analysis

Evaluation of matchmaking is based on relevance evaluation which concerns precision and recall of match results (Baeza-Yates and Ribeiro-Neto, 1999). Since our matchmaking process will return only the web services that can satisfy all capabilities requested in the query (i.e., they can match all relation expressions in the query but can also do more), this is called 'plug-in match' in Web Services Modelling Ontology (WSMO) (WSMO, 2004; Keller et al., 2004). As a result, we assume precision is 1 as all services returned are definitely relevant to the query. However, the recall value may be low as there may be some relevant services that are not returned because they match only some relation expressions of the query (at least one). This is called 'intersection match' in WSMO. We can apply intersection match to our matchmaking process instead and consider web services with intersection match as relevant to the query, so that the recall value will be increased. But by doing so, it is possible that a web service with plug-in match may be ranked lower than another web service with intersection match, because the former may match all requested capabilities but with low scores whereas the latter may match only some of the requested capabilities but with high scores. The web service with plug-in match may also be shifted down to lower sub-rank within the same rank order by the presence of another web service with intersection match which has the same match pattern score. The shift-down means the possibility that a service consumer will prefer and select the web service with intersection match instead of the one with plug-in match is high.

An experiment is conducted to study the effect of the shift-down. The matchmaking is tested under two scenarios

- when plug-in match is used
- when intersection match is used.

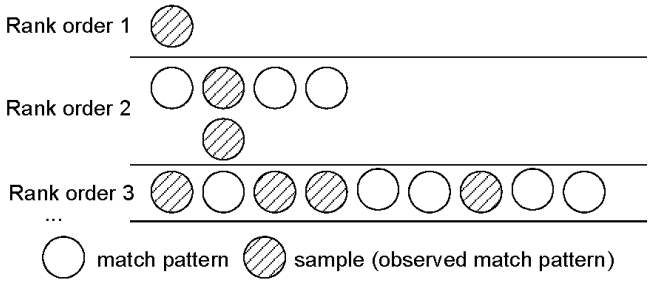
We consider the case of the query $\mathbb{Q}$ with six relation expressions and the match preference as well as the priority settings as in Section 6. There are 192 possible match patterns in total, with 11 rank orders. We select 50 match patterns out of 192 as the samples for observing their shift-down behaviour. These 50 match patterns are selected from each of the 11 rank orders in such a way that the 50 samples would reside in all rank orders in a normal distribution (Weis, 2004).

We start with the plug-in match scenario first. For each of the 50 sample match patterns, we compute a *relative distance* which reflects approximately how further down the sample is ranked, in relation to the match pattern at the top rank order. Since there may be multiple samples at a particular rank order, we compute an average of their relative distance values to represent a relative distance of any sample at that rank order. Suppose that we have the rank orders with match patterns assigned to each of them as in Figure 6. Some of the match patterns are the sample match patterns that we will observe. A relative distance of any samples at a particular rank order is computed by

RelativeDistanceOfSampleWithinRankOrder
= *the number of match patterns in the same rank order that are ranked higher*

RelativeDistanceOfAnySamplesAtRankOrder
= *the number of all match patterns in all higher rank orders*
+ *Average*(*RelativeDistanceOfSample WithinRankOrder*$_1$,…, *RelativeDistanceOfSampleWithinRankOrder*$_n$)
where n = the number of samples at that rank order.

Figure 6 Example of relative distance



RelativeDistanceOfAnySamplesAtRankOrder 1 = 0.

RelativeDistanceOfAnySamplesAtRankOrder 2 = 1 + Average(1,1) = 2.

RelativeDistanceOfAnySamplesAtRankOrder 3 = 6 + Average(0,2,3,6) = 8.75.

Determining a relative distance of any samples at a particular rank order helps simulate the shift-down effect. When the intersection match scenario is used, there will be more match patterns and these patterns will scatter in all rank orders and interleave with the match patterns of the plug-in scenario. Therefore, the position of a particular sample may shift down. When the relative distance of that sample within the rank order is increased and the number of match patterns in all higher rank orders is increased, the relative distance of any sample in that rank order is too.

We can repeat the calculation of relative distance of the 50 sample match patterns under the intersection match scenario and compares the result with that of the plug-in match scenario. Table 3 shows the comparison. The percentage of shift range is calculated from the change in relative distance when using intersection match compared to the relative distance under plug-in match scenario. Match patterns in higher rank order will be less affected by the intersection match, but for those in lower rank orders, the shift range increases exponentially. Thus the trade-off between recall value and relative distance should be considered. We can improve the situation by allowing service consumers to specify which of the requested relation expressions that must be matched (minimum requirement) in order to improve recall with less shift range.

The matchmaking process can also be improved in other aspects. Since ontological match is based on concept hierarchy, *specialised* match and *generalised* match at the moment cannot distinguish between matched concepts at different levels of the hierarchy. Matchmaking process can be refined to consider depth of concepts in the hierarchy as the concept that are closer to the concept specified in the query should be preferred. Also, in some cases, we may combine *specialised* match with *exact* match.

Measuring the performance of matchmaking process is beyond the scope of this paper. It is expected that the process will consume time to prepare and infer all ontologies that will be used, and compute possible match patterns, rank orders, and sub-ranking tables. Nevertheless, these can be performed at initialisation time prior to publishing and querying. Service providers and consumers should work conveniently on the pre-processed information. The idea is confirmed by (Srinivasan et al., 2004) which reports a performance evaluation of their ontology-based matchmaking process.

Table 3 Result of relative distance

Rank order	*Number of samples*	*Relative distance of any samples at rank order (plug-in match)*	*Relative distance of any samples at rank order (intersection match)*	*Shift range* (%)
1	1	0	0	0
2	2	2	2	0
3	4	10	10	0
4	6	27.33	28.83	5.49
5	7	56.86	65.43	15.08
6	10	93.1	123.1	32.22
7	7	130.86	204.43	56.22
8	6	157.17	308.83	96.50
9	4	178	449.75	152.67
10	2	188.5	630.5	234.48
11	1	191	802	319.90

Remark	*Number of relation expressions*	*Number of match pattern*	*Number of rank orders*	*Recall*
Plug-in	6	192	11	0.107 (192/1783)
Intersection	6	1783	22	1.00

7.2 Ranking capability analysis

By applying ordinal scale for ranking, the ranking is coarse-grained. As seen earlier, there are a number of match patterns that fall into the same ranking order. The ranking algorithm classifies them as equally ranked or 'unjudgeable' as it is not able to determine which one should be ranked higher or lower than another.

We are interested in the capability of the ranking algorithm. Ranking capability here refers to the ability of the algorithm to classify web services into different rank orders. We consider each pair of match patterns and if the algorithm can rank them, it has ranking capability over the pair. Intuitively, ranking capability can be determined by the proportion of the number of unjudgeable pairs of match patterns over the total number of match pattern pairs. This analysis can be performed on real integrated service profiles data, but in practice, the proportion may vary depending on the contents of the service profiles that are published. So it is difficult to realise the capability of the algorithm. In this paper, we instead simulate a ranking capability analysis on all possible match patterns that an integrated

service profile can match. Assume that service matches are evenly distributed; that is, for every possible match pattern, there is some service that matches the query by that pattern. Ranking capability can be computed by

$$\textit{Ranking capability} = 1 - \left(\left(\sum_{j=1}^{R} \sum_{i=1}^{P_j} (P_j - i) \right) \Big/ \sum_{k=1}^{P} (P - k) \right).$$

where

P_j: the number of match patterns in rank order j
R: the number of all possible rank orders
P: the number of all possible match patterns $= \sum_{j=1}^{R} P_j$.

From the formulae above, we can realise that

- $\sum_{i=1}^{P_j} (P_j - i)$ is the number of match pattern pairs within a rank order j, i.e., the number of unjudgeable pairs
- $\sum_{j=1}^{R} \sum_{i=1}^{P_j} (P_j - i)$ is the total number of unjudgeable match pattern pairs
- $\sum_{k=1}^{P} (P - k)$ is the total number of match pattern pairs.

We conduct an experiment in the most complex case where the largest number of match patterns will be involved. We assume the maximum number of match scores at four scales (i.e., 4, 3, 2, 1) and the match preference is set to the weakest level (i.e., 1). The experiment varies the number of the relation expressions that are specified in the query and needed to be matched. The result is in Table 4. With the number of relation expressions = 1, ranking capability is 1. This is obvious because each rank order will have only a single match pattern and therefore there is no unjudgeable pair within the same rank order. When the number of relation expressions increases, ranking capability is still satisfactory and getting closer to 1. In real situations where match distribution may not cover all rank orders, the result of ranking capability analysis may differ. Nevertheless, feature priority preference and profile priority preference should help refine ranking, and intuitively should improve ranking capability.

8 Semantic web services discovery architecture

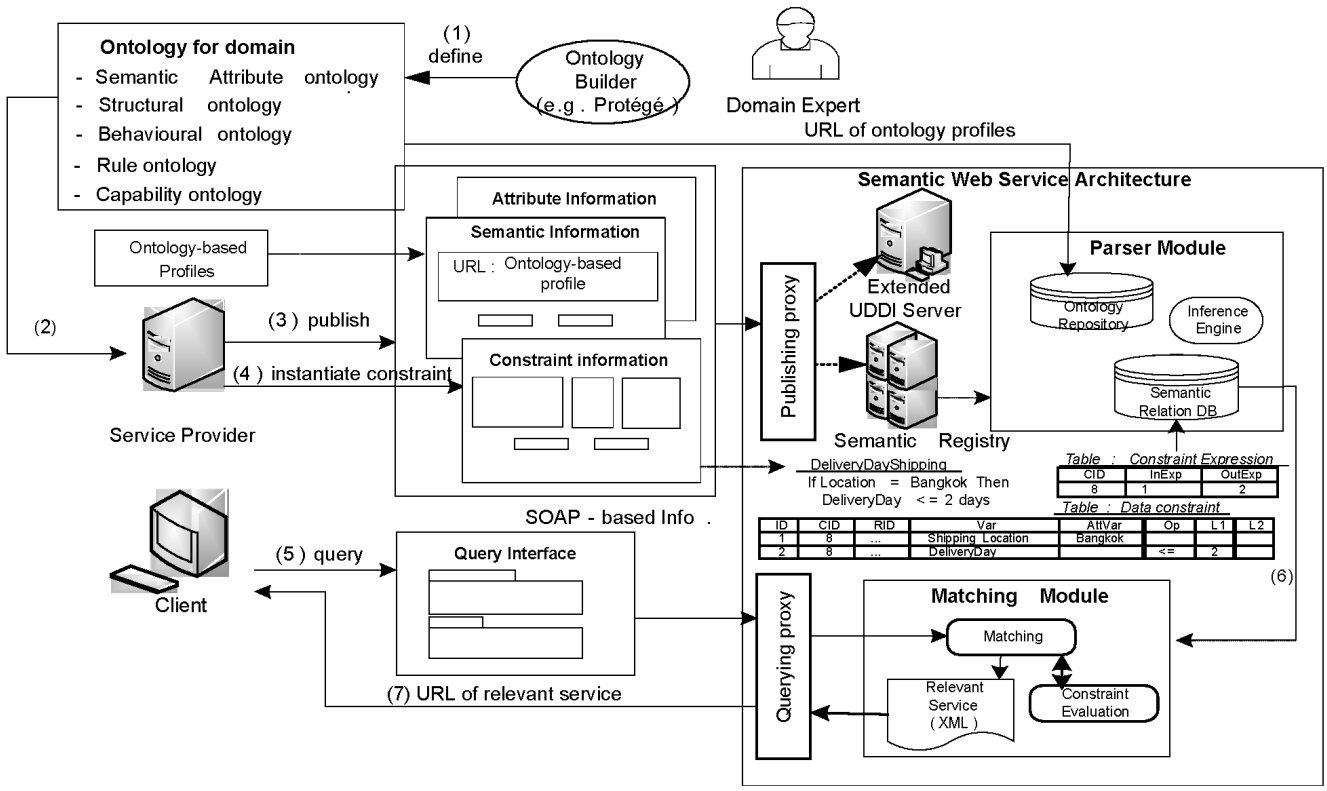
We design and implement the semantic web services discovery architecture depicted in Figure 7. The architecture integrates together the UDDI registry and a semantic registry. The UDDI registry here is extended to accommodate a variety of simple attributes that result from the survey (cf Table 1). The semantic registry will accommodate ontology-based profiles. Hence service consumers can have a mixture of the traditional way of attribute query and semantic query. All the upper ontologies proposed in this work will be stored in the ontology repository within the framework. Domain experts can load these upper ontologies in order to derive shared domain ontologies by using an ontology editor such as Protégé (Protégé, 2001) (1). Such domain ontologies may be stored somewhere in the network, e.g., at the domain experts' organisations. However, the domain experts are required to store the URLs of these domain ontologies with the ontology repository. This is for the semantic registry to be able to preprocess domain ontologies and perform reasoning, and also for service providers to load such domain ontologies in order to derive their ontology-based profiles (2). Via a GUI, service providers can publish their profiles through the publishing proxy (3) and instantiate service constraints (4). The publishing proxy will store simple attributes to UDDI, and generate all ontology-based profiles and store them in the ontology repository. The profiles are pre-processed to extract knowledge and reason further by the parser module which is integrated with an inference engine (e.g., Jena, 2003). The result is stored in the semantic relation database which is designed to handle facts from the profiles, more facts from inference, and service constraint expressions. Using a database is powerful as it can deal with huge information and its pre-processing helps prepare information for future retrieval.

The architecture can provide the service consumers with a GUI that corresponds to the ontologies of the domain so that the consumers can specify query onto the profiles more easily (5). The query will go through the querying proxy to the matching module. While performing matching, the matching engine may interact with the constraint evaluation engine. Constraint expressions in the database can be translated into other language such as RuleML (Iwaihara et al., 2002) or SWRL (Horrocks et al., 2003) and then a rule engine is used to evaluate them (6). The matching engine also performs ranking. Matched services will be ranked and reported in an XML document which will be returned to the consumer (7).

Table 4 Result from ranking capability experiment

Number of relation expressions	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
Number of match patterns	4	16	64	256	1024	4096	16384	65536
Number of rank orders	4	7	10	13	16	19	22	25
Number of unjudgeable pairs of match patterns	0	14	258	3918	57640	849770	1.264E7	1.895E8
No.of pairs of match patterns	6	120	2016	32640	523776	8386560	1.342E8	2.147E9
Ranking capability	1	0.883	0.872	0.879	0.889	0.898	0.905	0.912

Figure 7 Semantic web service discovery architecture



9 Related work

Describing service descriptions based on ontology specification is initially an effort by the DAML Services coalition. They propose DAML-S (The DAML Services Coalition, 2002) (now becomes OWL-S (Martin et al., 2004)), an ontology-based specification for service descriptions which consists of three profiles, i.e., Service Profile, Process Model, and Grounding Profile. Even so, only Service Profile is aimed for discovery. The Service Profile is defined in terms of functional attributes such as input, output, precondition, and effect, as well as some description that describes general attributes of the service such as provider information. The Service Profile overlaps by function with our attribute profile and behavioural profile. However, our attribute profile will accommodate more useful attributes as a result of an empirical survey, and we exercise discovery on all parts of the behavioural profile. The effort by (Paolucci et al., 2002) discovers web services based on operation, input, and output only; scale of matching by ontological subsumption is proposed but without ranking. In our approach, all aspects of the behavioural profile can be used in the query including precondition and effect. Our behavioural profile is also enhanced by the rule profile and service constraint evaluation. As suggested by (Trastour et al., 2002; Li and Horrocks, 2003), additional profiles such as those in our approach can be supplementary profiles to OWL-S Service Profile.

Some other work that tries to extend matchmaking on DAML-S such as (Bansal and Vidal, 2003) presents an algorithm that consider details in DAML-S Process Model for service matching based on input and output. Jaeger and Tang (2004) takes properties into account for matching of two profiles. Ranking is proposed based on subsumption of input, output, and property.

Recently, WSMO (2004) provides the conceptual framework for semantically describing web services. Based on WSMO, the Web Services Modeling Language (WSML) (Bruijn et al., 2005) implements this conceptual framework in a formal language for annotating web services with semantic information. WSML defines semantics in terms of four elements: ontologies, goals, web service descriptions, and mediators. Ontologies provide vocabularies, concepts, instances, and axioms that will be used by other elements. Goals are similar to queries. Web service descriptions describe capability in terms of assumption, precondition, postcondition, effect, and allow for interface and orchestration specifications. Mediators connect different WSMO elements and resolve heterogeneity between them. Although not the same, our integrated service profile has capabilities that correspond to many of WSML elements. Our semantic attribute profile and structural profile allows specifications of concepts and instances. The simple attribute profile allows for references to interface or orchestration specifications. The behavioural profile corresponds to the WSML capability. WSML provides syntax for conceptual structure of web services but does not say exactly what capability or vocabulary should be defined. Our integrated service profile gives clearer picture of service descriptions since the profile defines more concrete details, e.g., what attributes should be defined in the attribute profile, what basic information should be provided

in the structural profile, and the provision of shared domain ontologies which are concrete template for descriptions. WSML capability is defined by formal logical expressions, so it is powerful but would require complex reasoning for query and powerful tools. Matchmaking in WSMO defines types of matching based on the number of elements that are matched (i.e., exact, subsumption, plug-in, intersection). Ranking is simple and based on such types of matching. Our matchmaking and ranking is more complex and refined as there are several profiles involved and each profile will be considered for matching first before combining the results for ranking. Types of matching in our approach vary depending on the capabilities that are considered. Nevertheless, it is possible for WSMO to adopt our matchmaking and ranking approaches because of the similarities between the proposed profiles and WSML elements aforementioned. For example, our criteria and ordinal scale defined for matching ontological terms or sets of the terms may be applied to ontological concepts in WSML ontologies and web services descriptions. The criteria and matching scale for numerical constraints may also be applied to numerical constraint axioms. Consequently, the proposed ranking approach is applicable.

UDDI version 4 is trying to incorporate an ontology-based taxonomy for the standard categories of Business Entity and Business Service (Paolucci and Sycara, 2004). This effort will allow UDDI to also return businesses or services of a specialised or generalised category. This corresponds to having business or service categories defined as semantic attributes in our approach. However, the semantic attribute profile is open to any attributes.

Other work presents semantic-based frameworks based on description logics formalisation and description logic reasoning. The approaches by Trastour et al. (2002) and Li and Horrocks (2003) propose matchmaking in the e-commerce scenario in a multi-agent system when an advertisement represents the product description. This corresponds to the structural profile. They consider matching by subsumption relationship between the request and each advertisement in the repository. Ranking is not addressed in their work. Sycara et al. (2002) propose the agent-based framework in which the service capability is described by a description language LARKS. They propose a matchmaker which consists of a number of filters, each of which performs partial matching on the descriptions. Advertisements and request specifications will be compared whether they are sufficiently similar by using TF-IDF method and word distance values. Signature and constraints of input and output are also checked. Our work focuses on the specification of the profiles and mainly uses approximate match based on similarity through subsumption to determine the degree of similarity, as also adopted by (Paolucci et al., 2002; Di Noia et al., 2003; Andreasen et al., 2003).

Other work that integrates semantic information into UDDI architecture such as Sivashanmugan et al. (2003) enhances service descriptions by using ontological information to annotate service functions in the WSDL document. They use ontological subsumption to match operation, input, and output.

10 Conclusion

The contribution of this work is the integrated service profile which is a combination of the traditional attribute-based description and ontology-based specifications for use in matchmaking of web services. The integrated service profile considers many aspects of the service capability and is therefore richer for service providers to publish and for service consumers to query. The proposed matching scheme gives an intuitive ordinal scale based on ontological subsumption which considers semantic compatibility. The proposed ranking methodology can be performed across profiles and refined under user preference setting. An analysis on matchmaking and ranking processes gives some insight of the methodology and gives confidence over the practicality of the approach. Our integrated service profile is in accordance with the service-oriented model part of the web services architecture (Booth et al., 2004), which models a web service to have information about the provider, the syntax and semantics of the service, the tasks within the service, and a business policy.

Due to the richness of the profile, overheads exist when publishing and querying ontology specifications. However, the discovery architecture tries to facilitate in some way such as providing GUI that can guide providers and consumers to publish and query, the use of a powerful database to store facts, and preprocessing of knowledge extraction from the profiles.

The performance of the architecture will be evaluated in the future work. It is possible to incorporate distance-based matching on the concept hierarchy of the ontology with our approach so that ranking can be refined. More types of profile can be introduced to the integrated service profile such as the service composition profile.

Acknowledgement

This research is part of the Engineering New Paradigm Software for Enterprises with Service-Oriented Architecture Project, supported by Thailand's Software Industry Promotion Agency (Public Organisation).

References

Andreasen, T., Bulskov, H. and Knappe, R. (2003) 'On ontology-based querying', *Proceedings of Workshop on Ontologies and Distributed Systems, 18th International Joint Conference on Artificial Intelligence*, August.

Baader, F., Calvanese, D., McGuinness, D., Nardi, D. and Patel-Schneider, P.F. (2003) *The Description Logic Handbook: Theory, Implementation, and Applications*, Cambridge University Press, Cambridge.

Baeza-Yates, R. and Ribeiro-Neto, B. (1999) *Modern Information Retrieval*, Addison Wesley for ACM, MA.

Bansal, S. and Vidal, J.M. (2003) 'Matchmaking of web services based on the DAML-S service model', *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems*, *(AAMAS 2003)*.

Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C. and Orchard, D. (2004) *Web Services Architecture W3C Working Group Note 11 February 2004*, Available online at http://www.w3.org/TR/ws-arch/.

Bruijn, D.J., Lausen, H., Polleres, A. and Fensel, D. (2005) *The Web Service Modelling Language WSML: An Overview*, *DERI Technical Report 2005-06-16*, June.

Christensen, E., Curbera, F., Meredith, G. and Weerawarana, S. (2001) *Web Services Description Language (WSDL) 1.1*, Available online at http://www.w3.org/TR/wsdl/.

Constantinescu, I. and Faltings, B. (2003) 'Efficient matchmaking and directory services', *Proceedings of IEEE/WIC International Conference on Web Intelligence (WI'03)*, 13–17 October.

Di Noia, T., Di Sciascio, E., Donini, F.M. and Mongiello, M. (2003) 'A system for principled matchmaking in an electronic marketplace', *Proceedings of 12th International World Wide Web Conference (WWW 2003)*.

Dumas, M., O'Sullivan, J. and Heravizadeh, M. (2001) 'Towards a semantic framework for service description', *Proceedings of 9th International Conference on Database Semantics*, Hong Kong, April.

Gómez-Pérez, A. (1999) 'Ontological engineering: a state of the art', *Expert Update*, British Computer Society, Autumn, Vol. 2, No. 3, pp.33–43.

Gravano, L., Ipeirotis, P.G., Jagadish, H.V., Koudas, N., Muthukrishnan, S., Pietarinen, L. and Srivastava, D. (2001) 'Using q-grams in a DBMS for approximate string processing', *IEEE Data Engineering*, Vol. 24, No. 4, pp.28–34.

Gruber, T. (1993) 'A translation approach to portable ontologies', *Knowledge Acquisition*, Vol. 5, No. 2, pp.199–220.

Hay, D. and Healy, K.A. (2000) 'Defining business rules ~ what are they really?', *Technical Report 1.3*, The Business Rules Group, July, Available online at http://www.business rulesgroup.org/first_paper/br01c0.htm.

Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B. and Dean, M. (2003) 'SWRL: a semantic web rule language combining OWL and RuleML', Available online at http://www.daml.org/2003/11/swrl/.

Iwaihara, M., Kozawa, M., Narazaki, J. and Kambayashi, Y. (2002) 'A system for querying and viewing business constraints', Schroeder, M. and Wagner, G. (Eds.): *Proceedings of International Workshop on Rule Markup Languages for Business Rules on the Semantic Web*, Sardinia, Italy, 14 June.

Jaeger, M.C. and Tang, S. (2004) 'Ranked matching for service descriptions using DAML-S', *Proceedings of CAiSE Workshops*, Vol. 3, pp.217–228.

Jena (2003) *Semantic Web Framework: Jena*, Available online at http://jena.sourceforge.net/index.html.

Keller, U., Lara, R., Polleres, A., Toma, I., Kifer, M., Fensel, D. (2004) 'D5.1 v0.1 WSMO Discovery', *WSML Working Draft*, 13 September, Available online at http://www.wsmo.org/ 2004/d5/d5.1/v0.1/20040913/.

Larichev, O.I. (2001) 'Ranking multicriteria alternatives: the method ZAPROS III', *European Journal of Operation Research*, Vol. 131, No. 3, pp.550–558.

Li, L. and Horrocks, I. (2003) 'A software framework for matchmaking based on semantic web technology', *Proceedings of the 12th International World Wide Web Conference (WWW 2003)*.

Liskov, B.H. and Wing, J.M. (1994) 'A behavioural notion of subtyping', *ACM Transactions on Programming Languages and Systems*, Vol. 16, No. 6, pp.1811–1841.

Lynne, M. and Soh, C. (2002) 'Structural influences on global ecommerce activity', *Journal of Global Information Management*, Vol. 10, No. 1, January–March, pp.5–12.

Martin, D., Paolucci, M., McIlraith, S., Burstein, M., McDermott, D., McGuinness, D., Parsia, B., Payne, T., Sabou, M., Solanki, M., Srinivasan, N. and Sycara, K. (2004) 'Bringing semantics to web services: the OWL-S approach', *Proceedings of 1st International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004)*, July.

Navarro, G. (2001) 'A guided tour to approximate string matching', *ACM Computing Surveys*, Vol. 33, No. 1, pp.31–88.

Oaks, P., ter Hofstede, A.H.M. and Edmond, D. (2003) 'Capabilities: describing what services can do', *Proceedings of the 1st International Conference on Service Oriented Computing*, 15–18 December, Italy.

Pan, J.Z. and Horrocks, I. (2004) *OWL-E: Extending OWL with Expressive Datatype Expressions*, IMG Technical Report, 29 April.

Paolucci, M. and Sycara, K. (2004) *UDDI Spec TC V4 Proposal Semantic Search*, Available online at http://www.oasis-open.org/committees/uddi-spec/doc/req/ uddi-spec-tc-req029-semanticsearch-20040308.doc.

Paolucci, M., Kawamura, T., Payne, T.R. and Sycara, K. (2002) 'Semantic matching of web services capabilities', *Proceedings of 1st International Semantic Web Conference (ISWC 2002)*, LNCS, Vol. 2342, Springer Verlag.

Protégé (2001) Available online at http://protege.stanford.edu/.

Resnik, P. (1995) 'Using information content to evaluate semantic similarity in a taxonomy', *Proceedings of International Joint Conference on Artificial Intelligence*, November.

Sivashanmugan, K., Verma, K., Sheth, A. and Miller, J. (2003) 'Adding semantics to web services standards', *Proceedings of the International Conference on Web Services*, pp.395–401.

Sriharee, N. and Senivongse, T. (2003) 'Discovering web services using behavioural constraint and ontology', *Proceedings of 4th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS 2003)*, Paris, France, November, pp.248–259.

Sriharee, N. and Senivongse, T. (2005) 'Enriching UDDI information model with an integrated service profile', *Proceedings of 5th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS 2005)*, Athens, Greece, 15–17 June.

Sriharee, N., Senivongse, T., Teppaboot, C. and Futatsugi, K. (2004a) 'Adding semantics to attribute-based discovery of web services', *Proceedings of International Symposium on Web Services and Applications (ISWS' 04)*, Las Vegas, Nevada, USA, 21–24 June, pp.790–794.

Sriharee, N., Senivongse, T., Verma, K. and Sheth, A. (2004b) 'On using WS-policy, ontology and rule reasoning to discover web services', *Proceedings of the IFIP International Conference on Intelligence in Communication Systems (INTELLCOMM 04)*, 23–26 November, Bangkok, Thailand, pp.246–255.

Srinivasan, N., Paolucci, M. and Sycara, K. (2004) 'An efficient algorithm for OWL-S based semantic search in UDDI', *Proceedings of 1st International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004)*, San Diego, CA, USA, July 6.

Sycara, K., Widoff, S., Klusch, M. and Lu., J. (2002) 'LARKS: dynamic matchmaking among heterogeneous software agents in cyberspace', *Autonomous Agents and Multi-Agent Systems*, Vol. 5, No. 2, June, pp.173–203.

Tapabut, C., Senivongse, T. and Futatsugi, K. (2002) 'Defining attribute templates for descriptions of distributed services', *Proceedings of 9th Asia-Pacific Software Engineering Conference (APSEC 2002)*, Gold Coast, Australia, December, pp.425–434.

The DAML Services Coalition (2002) 'DAML-S web service description for the semantic web', *Proceedings of 1st International Semantic Web Conference (ISWC 2002)*, Sardinia, Italy, LNCS, Vol. 2342, Springer-Verlag.

Trastour, D., Bartolini, C. and Gonzalez-Castillo, J. (2001) 'A semantic web approach to service description for matchmaking of services', *Proceedings of the International Semantic Web Working Symposium (SWWS'01)*, pp.447–461.

Trastour, D., Bartolini, C. and Preist, C. (2002) 'Semantic Web Support for the Business-to-Business E-Commerce Lifecycle', *Proceedings of 11th International World Wide Web Conference (WWW 2002)*, pp.89–98.

uddi.org. (2002) *UDDI: Universal Description, Discovery, and Integration of Web Services*, Available online at http://www.uddi.org.

Ukkonen, E. (1992) 'Approximate string matching with q-grams and maximal matches', *Theoretical Computer Science*, Vol. 92, No. 1, pp.191–211.

W3C (2004) *OWL Web Ontology Language Overview*, Available online at http://www.w3.org/TR/2004/REC-owl-features-20040210/.

Weiss, N.A. (2004) *Introductory Statistics*, 7th ed., Addison Wesley, ISBN: 0201771314, May, Addison-Wesley, Lebanon, Indiana, USA.

Wickler, G. (1999) *Using Expressive and Flexible Action Representations to Reason about Capabilities for Intelligent Agent Cooperation*, PhD Thesis, University of Edinburgh, Edinburgh, Scotland.

WSMO (2004) *Web Services Modelling Ontology*, Available online at http://www.wsmo.org.

Zaremski, A.M. and Wing, J.M. (1997) 'Specification matching of software components', *ACM Transactions on Software Engineering and Methodology*, Vol. 6, No. 4, pp.333–369.