Discovering Semantic Web Services with Process Specifications

Piya Suwannopas and Twittie Senivongse

Department of Computer Engineering, Chulalongkorn University
Phyathai Road, Pathumwan, Bangkok 10330 Thailand
piya.su@student.chula.ac.th, twittie.s@chula.ac.th

Abstract. Service discovery is one of the crucial issues for service-oriented architectural model. Recently the trend is towards semantic discovery by which semantic descriptions are the basis for service matchmaking instead of simple search based on service attributes. OWL-S is a widely adopted semantic specification for Web Services which comprises three profiles. Among those, process model is the profile that describes dynamic behaviour of Web Services in terms of functional aspects and process flows, and is generally aimed for service enactment, composition, and monitoring. This paper presents a new approach to use OWL-S process model for service discovery purpose. A Web Service can have its internal process described as an OWL-S process model specification, and a service consumer can query for a Web Service with a particular process detail. Matchmaking will be based on flexible ontological matching and evaluation of constraints on the functional behaviour and process flow of the Web Service. The architecture for process-based discovery is also presented.

1 Introduction

Service discovery is an important part of service-oriented computing in which services, as building blocks for building applications, are provided and distributed in large-scale open environment [1]. Provided services will publish generalised descriptions of their capability to a matchmaker whereas service consumers consult the matchmaker to identify potential services that most closely satisfy their needs. The effectiveness of service discovery relies on the richness of service metadata and the matchmaking mechanism that utilises the expressiveness of the metadata. Current Web Services Standards realise this concept and provide UDDI [2] as a standard registry that performs matchmaking based on matching of syntactic service attribute values.

From our previous study [3], a service description model has been defined as a result of an empirical survey about service advertisements on the Internet (Fig. 1). The model shows that service advertisements should reflect different aspects of service capabilities; some are simple characteristics and may be in the form of simple attributes whereas some are more complex capabilities and require some specification languages to express them. (Those highlighted in Fig. 1 have no correspondences in

UDDI.) This model is generic, meaning that it is independent from any specific representation languages and can be used simply for information or for other purposes such as automatic service discovery or composition.

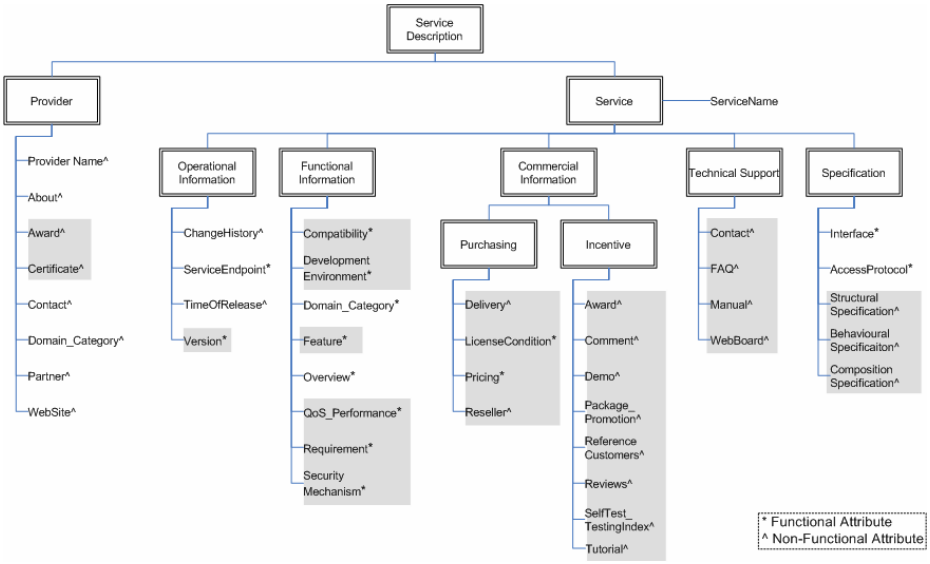

Fig. 1. Service description model from survey [3].

One way to enrich service metadata is by using ontology languages to represent service descriptions. This approach is gaining a lot of attention in Web Services community as ontology languages are expressive for describing several aspects of service capabilities and ontological reasoning also provides a way to infer more about the capabilities. Semantic Web Services are Web Services in which ontologies ascribe meanings to published service descriptions so that software systems representing prospective service consumers can interpret and invoke them [4]. With this vision, the Web Ontology Language for Services (OWL-S) consortium contributes with an OWL-S specification [5] which is the building block for encoding rich semantic service descriptions in a way that builds naturally upon OWL language. OWL-S consists of three profiles, namely service profile, process model, and service grounding. Service profile defines basic and functional properties of the service as well as functional behaviour. Process model details service operation in terms of functional behaviour, control structure, and data flow structure required to execute the service. Service grounding specifies details of how to access the service by mapping from an abstract service specification (process model) to concrete specification (WSDL). It can be seen that OWL-S and the model in Fig. 1 share some characteristic; they both model services with simple attributes and more complex specifications.

Our previous work [6] proposes an integrated service profile that corresponds to the model in Fig. 1. The integrated service profile is a collection of ontology-based profiles for services, including the attribute, structural, behavioural, and rule profiles,

and it overlaps with OWL-S. This paper extends the integrated service profile with the focus on the composition specification of a service. Composition specification shows how simple components are composed into a service and may be expressed as a hierarchy of goal and subgoals or as a workflow of tasks for service execution [1]. This paper is interested in describing the composition specification as a workflow and we borrow OWL-S process model to represent the workflow specification.

OWL-S process model is found in use by researches in service composition and workflow coordination and monitoring, but it can also be used for in-depth analysis for matchmaking to see whether the service meets process constraints required by the service consumer. This is to check a dynamic aspect of the service. For example, the service consumer may want to find a software store with a workflow such that, after processing the purchase order of the customer, the store registers the customer for the software training programme. The store service with such automatic registration for training should be preferable to ones without training. Sometimes the flow may have a constraint such that automatic training registration is available only if the purchase is worth more than 0.5 million bahts (Thailand currency). Such a constraint will have to be taken into account during matchmaking. Here we present an example of the services using OWL-S process model to describe their internal processes. A service consumer can issue a process-based query. The services are queried on their functional behaviour and flow of their process. Ontological reasoning and evaluation of the rule-based constraints on the behaviour and process flow are considered.

The rest of the paper starts with Section 2 that discusses related work. Section 3 outlines the constructs of OWL-S process model for process specification. Section 4 gives an example of the process specifications of three services described using OWL-S process model. Matching criteria are summarised in Section 5 and used in Section 6 to consider matching for a query. Section 7 presents a process-based discovery framework and Section 8 concludes the paper.

2 Related Work

Semantics-based service discovery is accomplished mainly by the use of ontology to describe service capabilities. Web Services Modeling Ontology (WSMO) [7] provides a framework for describing semantic Web Services with Web Services Modeling Language (SWML) [8] as a formal language that realises the framework. 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. As WSMO shares with OWL-S the vision that ontologies are essential to support automatic discovery, it is possible for our work to adopt either of their process-related specifications. However, at the moment OWL-S can be implemented without stipulating framework and several tools exist. We adopt OWL-S process model for process specification in this paper.

Most of research work in service discovery area focuses on search based on a particular aspect of the service and little is found to concentrate on process-based discovery. UDDI version 4 is incorporating an ontology-based taxonomy for the standard categories of Business Entity and Business Service entries that are registered with UDDI [9]. This will allow UDDI to be able to look for the businesses or services of a specialised or generalised category. The work in [10] shows how ontology describing general knowledge of a particular service domain can be used for search. The work in [11], [12] focuses on searching functional behaviour but they do not consider search with behavioural constraints. In [13], an efficient search algorithm is devised for services described by OWL-S but the search considers only the OWL-S service profile. In [14], process ontology is used as a basis for service discovery. The process ontology is described by the service process, constituent subtasks, connection ports between subtasks and connection mechanisms, and exceptions within the process. The query is done by a PQL language. Unlike our approach, the process ontology in this work follows the goal-subgoal model of service composition, not the workflow model, and it does not accommodate for process constraints.

Service discovery and service composition share a characteristic such that both aim to identify services that can satisfy users' requirements. Nevertheless, service discovery tends to identify individual services that can answer to a particular query, whereas service composition identifies a group of services that can work together to satisfy a certain goal. In the area of Web Service composition, OWL-S process model is used in several researches for describing Web Services. In [15], an AI planner called OWLS-Xplan is proposed to compose Web Services. An OWL-S process model is used to specify input, precondition, output, and effect of the goal (i.e. the composite service) and of the individual Web Services to be composed. The goal in OWL-S process model will be translated into a planning domain description in PDDL in order for the planner to generate a plan sequence as a workflow of individual Web Services. The work in [16] integrates an OWL reasoner with an AI planner and shows how OWL or SWRL [17] is used to encode the preconditions and effects of the Web Services in the composition process. The Web Services are also described by OWL-S process model. By using OWL, the composition gains the reasoning power of OWL in the evaluation of the preconditions and the update of the effects that have impacts on real world knowledge. Although these researches above conduct some analysis on OWL-S process model, they concern the functional behaviour part of the process model in service composition. In our work, we focus on analysing not only the functional behaviour part but also the workflow part of individual Web Services in order to find any single services that can satisfy the query.

3 OWL-S Process Model

This section briefly describes the constructs of OWL-S process model that are of interest to this paper. A particular service is described by a service model and a process is a subclass of the service model. Fig. 2 shows OWL-S process ontology [5] with the classes and properties that altogether describe how a service works. A process describes its functional behaviour by specifying inputs, outputs, preconditions,

and effects (IOPE) of its performance. As the name implies, a precondition is a logical expression which must hold for the process to be successfully invoked. Local refers to an auxiliary parameter that is bound to the precondition and is useful for determining the logical value of the precondition. Result refers to a coupled output and effect and can be constrained by an incondition property which specifies the logical condition under which the result occurs; hence the corresponding output and effect become conditional output and conditional effect. Result variable is also an auxiliary parameter that is bound to a result and useful for determining the associated incondition.

The process is further described as a composition of subprocesses. The subprocess can be atomic, composite, or simple process. An atomic process is one which has no further subprocesses, is directly invocable, and executes in a single step. A composite process is decomposed into other non-composite or composite processes. The decomposition can be specified by using control constructs, i.e. sequence, split, split-join, any-order, choice, if-then-else, iterate, repeat-while, repeat-until. A simple process is an abstraction that provides a view of some atomic process or a simplified representation of some composite process and is not invocable.

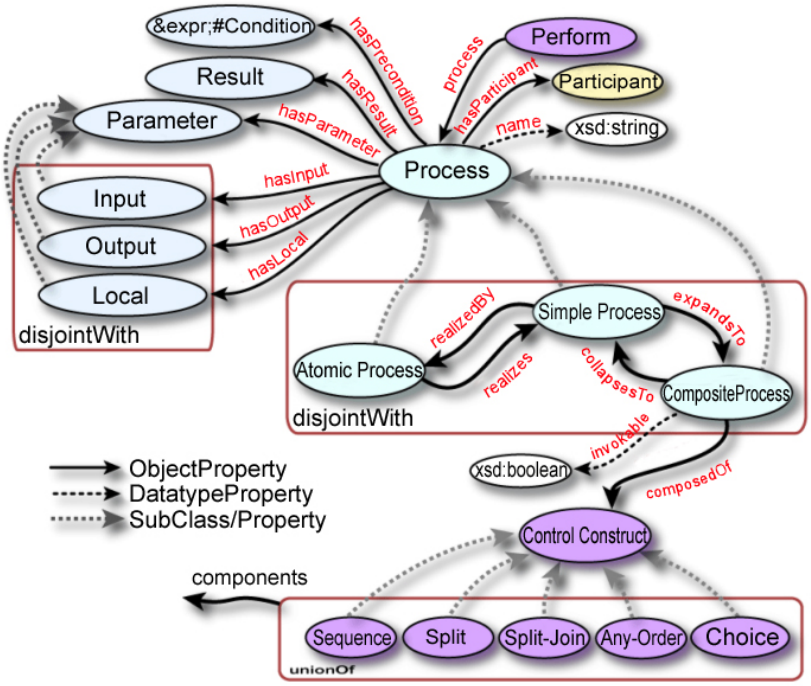

Fig. 2. Top level of process ontology [5]

Since the constraints in OWL-S process model – either the preconditions, the conditions of the results, or the guards on the control flow – are represented as logical formula, these logical expressions are treated as literals – either XML literals or string

literals. Therefore several languages can be used to express these constraints (e.g. SWRL, RDF, KIF, PDDL). In this paper we represent such process constraints with SWRL rule expressions.

4 Process Specifications

Bank loan service is used as an example for process-based discovery. Fig. 3 shows the first part of the process specification of a loan service S_1 written in OWL-S process model. This part describes the functional behaviour of S_1.

```
1.  <process:CompositeProcess rdf:ID="LoanService">
2.    <process:hasInput>
3.      <process:Input rdf:ID="CustomerInfo"/>
4.    </process:hasInput>
5.    <process:hasOutput>
6.      <process:Output rdf:ID="LoanInterestRate"/>
7.    <process:hasOutput/> …
8.    <process:hasLocal rdf:resource="#IncomePerMonth"/>
9.    <process:hasPrecondition>
10.     <expr:SWRL-Condition rdf:ID="IncomeCondition">
11.       <expr:expressionLanguage rdf:resource="&Expression.owl#SWRL"/>
12.       <expr:expressionBody rdf:datatype="Literal">
13.         swrlb:greaterThanOrEqual(#IncomePerMonth,10000) →
14.         hasIncomeStatus(#ValidIncome,"xsd:True")
15.       </expr:expressionBody>
16.     </expr:SWRL-Condition>
17.   </process:hasPrecondition>
18.   <process:hasResultVar rdf:resource="#LoanAmount"/>
19.   <process:hasResult>
20.     <process:Result rdf:ID="PremiumLoanResult">
21.       <process:inCondition>
22.         <expr:SWRL-Condition rdf:ID="PremiumLoanCondition">
23.           <expr:expressionLanguage rdf:resource="&Expression.owl#SWRL"/>
24.           <expr:expressionBody rdf:datatype="Literal">
25.             swrlb:greaterThan(#LoanAmount,300000) →
26.             hasPremiumLoanStatus(#PremiumLoanStatus,"xsd:True")
27.           </expr:expressionBody>
28.         </expr:SWRL-Condition>
29.       </process:inCondition>
30.       <process:hasEffect>
31.         <expr:SWRL-Condition rdf:ID=”PremiumCreditCardCondition”>
32.           <expr:expressionBody rdf:datatype="Literal">
33.             → chargedPremiumCreditCard(#LoanService, #PremiumCreditCardFee)
34.               swrlb:equal(#PremiumCreditCardFee, 0)
35.           </expr:expressionBody>
36.         </expr:SWRL-Condition>
37.       </process:hasEffect>
38.     </process:Result>
39.   </process:hasResult>
40.   <process:hasResult>
41.   <process:Result rdf:ID="NormalLoanResult">
42.       …
```

Fig. 3. Functional behaviour of S_1 in OWL-S process model

From the figure, the service requires customer information as an input (line 2-4), and gives loan interest rate as an output (line 5-7). The service has a precondition such

that the consumer needs to have income at least 10,000 bahts per month in order to use the service (line 9-17). The effects of this service are conditional, depending on the loan amount. If the loan is more than 300,000 bahts, it is a premium loan (line 21-29) and the consumer is entitled to apply for a premium credit card. This effect is further constrained by the annual credit card fee which is equal to 0 (line 30-37). On the other hand, if the loan is not more than 300,000 bahts, it is a normal loan (line 41) and the credit card effect will be subject to the annual fee. Note that all the constraints are expressed as SWRL rules.

The second part of the process specification of S_1 involves its workflow. This is depicted in Fig. 4. Suppose, in general, a loan service is composed of several classes of loan approval. Department approval process is performed when the loan amount is small or the loan is not critical and the decision can be made by the loan department manager. Branch approval process is performed when the loan is more critical but the decision can still be made within the branch by the branch manager. Otherwise the loan application has to be approved at the head quarter. The bank will maintain loan history of the customers for future reference.

Fig. 5 shows a snippet of OWL-S process specification for Fig. 4. The first guard condition checks whether the loan amount is less than or equal to 1 million bahts (line 64-72). The second guard condition determines whether the purpose of loan is for real estate (line 88-96).

For further comparison, we assume there are two more candidate services S_2 and S_3. These two services exhibit the same functional behaviour as S_1 (c.f. Fig. 3) but they have a slightly different workflow as in Fig. 6 and Fig. 7 respectively.

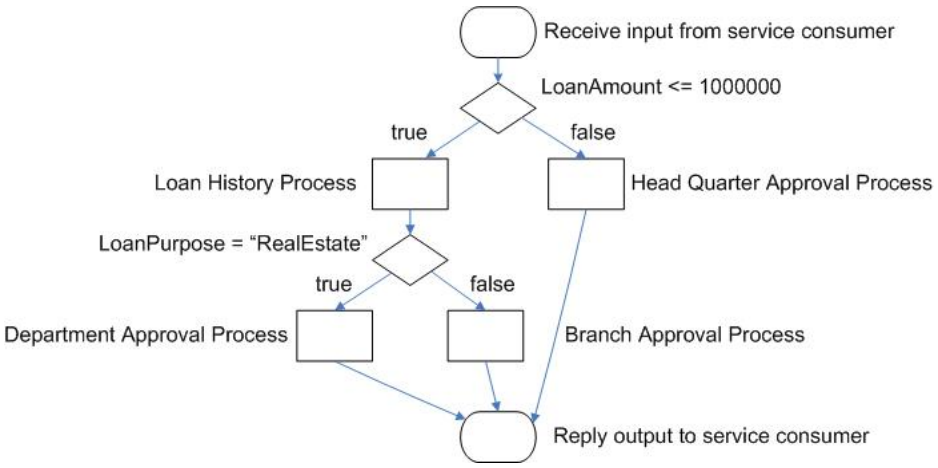

Fig. 4. Process flow of service S_1

```
<process:composedOf>
 <process:If-Then-Else rdf:ID="LoanAmount_If-Then-Else">
  <process:ifCondition>
   <expr:Condition rdf:ID="LoanAmountCondition">
    <expr:expressionLanguage rdf:resource="&Expression.owl#SWRL"/>
    <expr:expressionBody rdf:datatype="Literal">
     swrlb:lessThanOrEqual(#LoanAmount,1000000) →
     hasLoanAmountStatus(#SmallLoanAmount,"xsd:True")
    </expr:expressionBody>
   </expr:Condition>
  </process:ifCondition>
  <process:then>
   <process:Sequence rdf:ID="Bank_Sequence">
    <process:components>
     <process:ControlConstructList rdf:ID="LoanHistory_ControlConstructList">
      <list:first>
       <process:Perform rdf:ID="LoanHistoryPerform">
        <process:process>
         <process:AtomicProcess rdf:ID="LoanHistoryProcess"/>
        </process:process>
       </process:Perform>
      </list:first>
      <list:rest>
       <process:ControlConstructList rdf:ID="Bank_ControlConstructList">
        <list:first>
         <process:If-Then-Else rdf:ID="Purpose_If-Then-Else">
          <process:ifCondition>
           <expr:Condition rdf:ID="PurposeCondition">
            <expr:expressionLanguage rdf:resource="&Expression.owl#SWRL"/>
            <expr:expressionBody rdf:datatype="Literal">
             swrlb:equal(#LoanPurpose,"RealEstate") →
             hasPurposeStatus(#RealEstatePurpose,"xsd:True")
            </expr:expressionBody>
           </expr:Condition>
          </process:ifCondition>
          <process:then>
           <process:Sequence rdf:ID="Department_Sequence">
            <process:components>
             <process:ControlConstructList rdf:ID="Department_ControlConstructList">
              <list:first>
               <process:Perform rdf:ID="DepartmentApprovalPerform">
                <process:process>
                 <process:AtomicProcess rdf:ID="DepartmentApprovalProcess"/>
                </process:process>
               </process:Perform>
                    …
```

Fig. 5. Process flow of S_1 in OWL-S process model

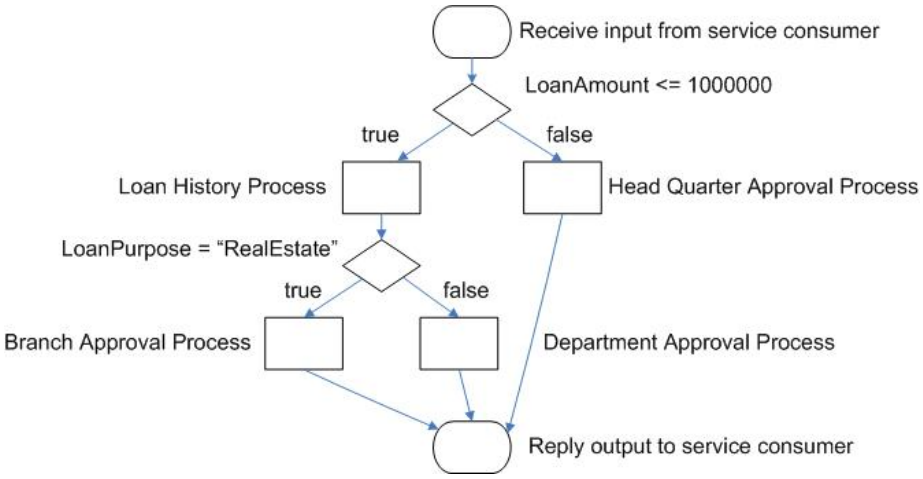

Fig. 6. Process flow of service S_2

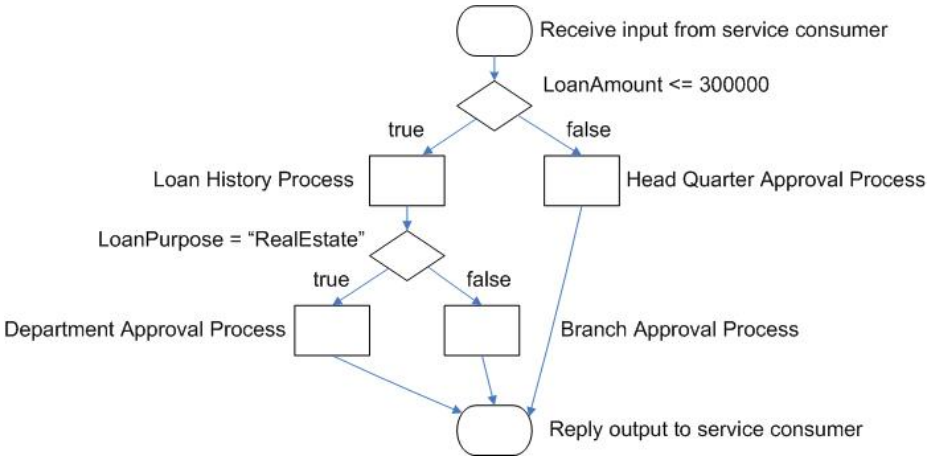

Fig. 7. Process flow of service S_3

5 Matching Criteria

To determine whether a process specification of a service can fulfill a service consumer's needs, matchmaking will perform ontological matching on the concepts within the specification and evaluate constraints on the functional behaviour and the guards on the control constructs in order to determine the actual behaviour of the service. Several matching criteria are defined:

5.1 Matching Ontological Concepts

Matching by subsumption and equivalence is the basis for matching ontological concepts in the query and the process specification. This approach is based on the IS-A taxonomy of the concepts shared within the service domain and has been adopted in literature including [10], [18], [6].

Let C_Q be the concept specified in the query and C_P be the concept in the process specification:

(i) *If* $C_Q \equiv C_P$ *then* C_P *is an exact match for* C_Q, where $\equiv$ means is equivalent to.

(ii) *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).

(iii) *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 process specification.

(iv) *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 process specification to be a match for the concept in the query provided that the two concepts have common characteristics through a common parent concept.

(v) *If none of the above relationships exist then* C_P *is a failed match for* C_Q.

5.2 Matching Numerical Ranges

Matching two numerical ranges compares the ranges of the possible values that are defined in the constraints. The degree of matching for numerical ranges can be determined as described below.

Let N_Q be a nonempty set of numerical range values of the expression in the query (E_Q), and N_P be a nonempty set of numerical range values of the expression in the process specification (E_P):

(i) *If* $N_P \subseteq N_Q$ *then* E_P *is an exact match for* E_Q

(ii) *If* $N_Q \subseteq N_P$ *then* E_P *is a plug-in match for* E_Q

(iii) *If* $(N_P \cap N_Q \neq \phi) \wedge (N_P \not\subseteq N_Q) \wedge (N_Q \not\subseteq N_P)$ *then* E_P *is a weak match for* E_Q

(iv) *If* $N_P \cap N_Q = \phi$ *then* E_P *is a failed match for* E_Q

5.3 Matching Logical Constraint

The service will match to the query if, by applying a set of values obtained from the query into the rule expression, the rule evaluation hits and returns true as a result. The expression in the head atom of the rule may be a numerical constraint or constraint on some data values, and these may require ontological reasoning, numerical computation, and also rule reasoning. We consider a match only when such evaluation returns true.

5.4 Matching Process Model

To check whether a process specification satisfies the query, we consider matching on all aspects of the functional behaviour and the processes within the workflow. For each aspect, it may need to perform ontological matching (Section 5.1) before considering other kind of constraint matching (Sections 5.2-5.3). The process specification will match the query if it satisfies the following:

(i) *input, unconditional output, unconditional effect, and process without guard satisfy ontological match in Section 5.1, and*

(ii) *precondition, conditional output, conditional effect, and process with guard satisfy relevant matching criteria in Section 5.2-5.3*

In other words, let $\mathbb{R}_Q$ and $\mathbb{R}_P$ be the sets of functional behaviour and workflow processes (with and without constraints) within the query and the process specification respectively:

ProcessModelMatch($\mathbb{R}_Q$, $\mathbb{R}_P$) = true $\Leftrightarrow$

$(\mathbb{R}_Q \subseteq \mathbb{R}_P) \wedge (\forall i, \exists j : (i \in \mathbb{R}_Q) \wedge (j \in \mathbb{R}_P) \wedge (i \; \Theta \; j))$

where Θ means having a kind of match as in Sections 5.1-5.3.

6 Process-Based Discovery

Assume a service consumer wants to apply for a 400,000-baht loan with a bank in order to buy a house. The consumer wants the bank that allows a loaner to apply for a credit card with no annual fee and approve the loan application at loan department level. This is to ensure that the loan process is quick. The consumer earns 20,000 bahts a month.

We present a query ($\mathbb{Q}$) as a collection of relation expressions. A relation expression is in the form of *property(subject, object)* which corresponds to an RDF statement <*subject, property, object*>. For a constraint that relates to a numerical value, such numerical constraint is represented as *property(argument, relationaloperator,*

literalvalue1, [literalvalue2,] unit). For the example above, the relation expressions are superscripted by symbols *C, E, G, and P* which refer to precondition, effect, guard, and process respectively:

$\mathbb{Q}$ ={hasIncomePerMonth(IncomePerMonth, 20000)^{C} ,
hasPremiumCreditCardFee(PremiumCreditCardFee, Equal, 0, baht)^{E} ,
hasLoanAmount(LoanAmount, 400000)^{G} ,
hasLoanPurpose(LoanPurpose, Housing)^{G} ,
hasProcess(Process, DepartmentApprovalProcess)^{P} }

To determine whether a service is a match, its process specification will also be treated as a collection of relation expressions in order to check against the set of relation expressions of the query. The rule expressions embedded in the process specification will be extracted and translated into a rule language in order to use a rule reasoning engine to check whether the rule is satisfied. In our implementation, SWRL rule will be translated into Jess script in order to use Jess engine [19].

If we look at S_1 and the query, to check whether the precondition holds for the query, we use the criterion to match numerical ranges (Section 5.2) and the consumer's income is an *exact match* and hence valid to use the service. To check the effect, we have to determine what S_1 will give as an effect since it is conditional. We first check the incondition by using matching of numerical ranges on the loan amount and the premium credit card effect is satisfied with an *exact match*. Then we use again the numerical range matching criterion to check whether the premium credit card offers 0 baht annual fee. This also returns an *exact match*. When all aspects of the functional behaviour of S_1 match to the query, S_1 is a potential service but we have to check further on its process flow. (In this example, the functional behaviour of S_2 and S_3 also matches to the query because we assume earlier that all three services exhibit the same functional behaviour.)

To consider the workflow of the service, we associate each process with guards that determine its performance. For example, the rules for all approval processes within the process specification of S_1 are listed below:

!hasLoanAmount(LoanAmount, LessThanOrEqual, 1000000, baht) →
    hasProcess(Process, HeadQuarterApprovalProcess);
hasLoanAmount(LoanAmount, LessThanOrEqual, 1000000, baht) →
    hasProcess(Process, LoanHistoryProcess);
hasLoanAmount(LoanAmount, LessThanOrEqual, 1000000, baht),
    hasLoanPurpose(LoanPurpose, RealEstate) →
    hasProcess(Process, DepartmentApprovalProcess);
hasLoanAmount(LoanAmount, LessThanOrEqual, 1000000, baht),
    !hasLoanPurpose(LoanPurpose, RealEstate) →
    hasProcess(Process, BranchApprovalProcess);

To check whether S_1 performs the requested process under the context of a particular query, we check whether the associated guards fire. This is possible when the information necessary for evaluating the guards can be obtained from the service consumer or from the process specification itself. In this example, the consumer requests for a department approval process. The first guard on loan amount fires with *exact match* by considering numerical ranges matching against the loan amount of the

consumer. For the second guard on loan purpose, we first use ontological matching (Section 5.1) to check the ontological value RealEstate. Assume that there is a domain ontology which defines an IS-A taxonomy for RealEstate with subconcepts such as Housing and Land. S_1's purpose will be a *generalised match*, and by matching logical constraints (Section 5.3), this second guard will also fire. Therefore, S_1 will perform department approval process under the constraints placed by the query. When S_1 matches with all aspects defined in the query, it will be returned as a match to the consumer. With this approach, S_2 will fail to match the query because the consumer's loan purpose will not cause the loan purpose guard associated with its department approval process to fire. Similarly, S_3 will also fail to match the query because the consumer's loan amount does not satisfy the loan amount guard associated with its department approval process.

Process-based discovery is effective when a shared process ontology of a particular service domain is assumed. The shared process ontology defines common pattern of the process within a domain which includes internal tasks and relevant conditions. This approach is possible as the concept of business process patterns exists [20], [21]. Service providers should publish process specifications that are derived from the domain process ontology, and service consumers should have some knowledge about the behaviour and workflow of the domain in order to compose an effective query. In our example, it should be commonly known that a bank loan process usually involves several classes of approval, and factors that influence the approvals include loan amount, loan purpose, and earning capability of the loaner. Although this process is internal to the bank, it is not classified business information since bank staff would normally give such information to the loaners. With a shared process ontology, the service consumer can submit a query without having to know other details of the candidate Web Services which may be considered as classified business rules; in our case, the service consumer does not need to know that the bank with a process specification such as S_1 has set a boundary of 1 million bahts for a head quarter approval. Process specifications are maintained by service providers; our approach does not require service consumers to have access to them.

7 Discovery Framework

The agent-based discovery framework in our previous work [6] is extended to accommodate process-based discovery. We develop the components within the architecture in Fig. 8 while also adopting existing ontology-based tools and rule engine.

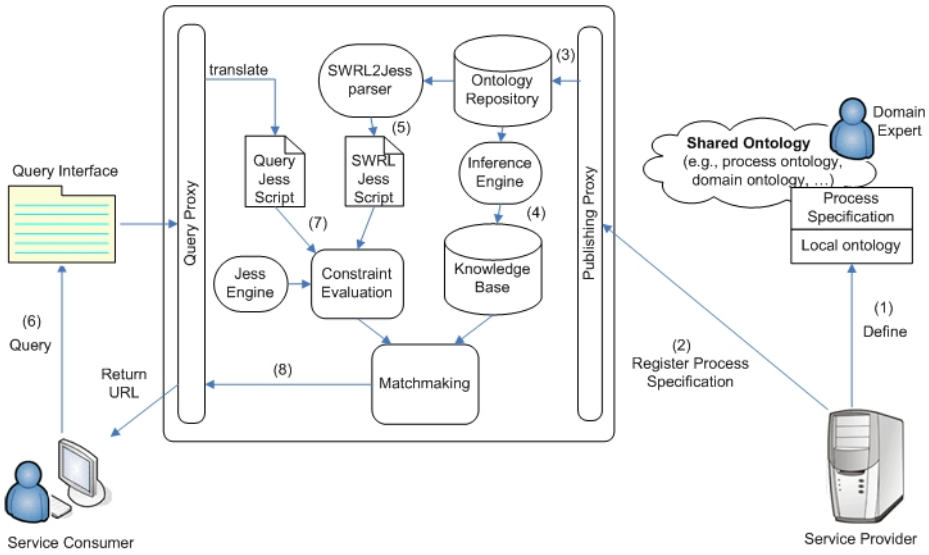

Fig. 8. Process-based discovery framework

In the figure, a service provider will define the process specification of the service as well as any necessary local ontology (1), using an ontology editor (e.g. Protégé). The definition may be based on shared ontology of the domain, which is defined by service domain experts. The service provider maintains the process specification and the local ontology, but also registers the specification with the agent via the publishing proxy (2). The publishing proxy will store the URL of the process specification and local ontology in the ontology repository (3). The agent may preprocess to extract knowledge and to reason from the shared ontologies prior to the matchmaking by using an inference engine (e.g. Jena [22]); the results are stored in a knowledge base (4). At discovery time, the process specification will be processed and rule constraints are extracted and translated into a rule script by a parser (i.e. SWRL2Jess parser) (5). The agent can provide the service consumers with a GUI template that corresponds to the process ontology of the domain so that the consumers can specify query onto the process specifications more easily (6). Internally, the query will be translated into RDF-based relation expressions and will pass through the query proxy. Rule constraints in the query are translated into a rule script so that it is evaluated against constraints in the process specification (7). The constraint evaluation module is integrated with a rule engine (e.g. Jess engine). Matchmaking module considers matching criteria and reports the result in an XML document which will be returned to the consumer (8).

8 Conclusion

We present a new approach to service discovery by using OWL-S process model to model functional behaviour and workflow of the services and querying on such process specifications. Constraints can be placed on the functional behaviour and guard the flow of process execution. Matchmaking uses ontological reasoning and constraints evaluation to determine the actual behaviour of the services. Service consumers can then look for the services with a satisfied internal process.

The example in this paper shows a query concerning if-then-else and sequence constructs. Query based on other constructs is also meaningful and possible. We are in the process of finishing the integration of process-based discovery with the framework in [6] so that the integrated service profile is more complete and fits well with the service description model in Fig. 1.

References

1. Huhns, M. N., Singh, M. P.: Service-Oriented Computing: Key Concepts and Principles. IEEE Internet Computing. January-February (2005) 75-81
2. uddi.org: UDDI: Universal Description, Discovery, and Integration of Web Services (Online). (2002). http://www.uddi.org
3. Tapabut, C., Senivongse, T., Futatsugi, K.: Defining Attribute Templates for Descriptions of Distributed Services. In: Proceedings of 9th Asia-Pacific Software Engineering Conference (APSEC 2002), Gold Coast, Australia, December (2002) 425-434
4. Burstein, M. et al.: Semantic Web Services Architecture. IEEE Internet Computing. September-October (2005) 72-81
5. OWL-S Coalition. OWL-S 1.1 Release (online). http://www.daml.org/services/owl-s/1.1/
6. Sriharee, N., Senivongse, T.: Matchmaking and Ranking of Semantic Web Services Using Integrated Service Profile. To be published in International Journal of Metadata, Semantics and Ontologies, Vol. 1, No. 2, Inderscience Publishers
7. WSMO. Web Services Modeling Ontology (online). (2004). http://www.wsmo.org
8. Bruijn, D.J., Lausen, H., Polleres, A., Fensel, D.: The Web Service Modeling Language WSML: An Overview. DERI Technical Report, June 16 (2005)
9. Paolucci, M., Sycara, K.: UDDI Spec TC V4 Proposal Semantic Search (online). (2004). http://www.oasis-open.org/committees/uddi-spec/doc/req/uddi-spec-tc-req029-semanticsearch-20040308.doc
10. Trastour, D., Bartolini, C., Gonzalez-Castillo, J.: A Semantic Web Approach to Service Description for Matchmaking of Services. In: Proceedings of the International Semantic Web Working Symposium (SWWS'01) (2001)
11. Paolucci, M. et al.: Semantic Matching of Web Services Capabilities. In: Proceedings of the 1st International Semantic Web Conference (ISWC 2002), Sardinia (Italy), Lecture Notes in Computer Science, Vol. 2342. Springer Verlag (2002)
12. Sivashanmugan, K., Verma, K., Sheth, A., Miller, J.: Adding Semantics to Web Services Standards. In: Proceedings of the International Conference on Web Services (2003)
13. Srinivasan, N., Paolucci, M., Sycara, K.: An Efficient Algorithm for OWL-S Based Semantic Search in UDDI. In: Proceedings of 1st International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004), San Diego, CA, USA, July 6, (2004)

14. Klein, M., Bernstein, A.: Searching for Services on the Semantic Web Using Process Ontologies. The Emerging Semantic Web – Selected papers from 1st Semantic Web Working Symposium. I. Cruz et al. (Eds.) IOS press, Amsterdam (2002) 159-172
15. Klusch, M., Gerber, A., Schmidt, M.: Semantic Web Service Composition Planning with OWLS-Xplan. In: Proceedings of 1st Intl. AAAI Fall Symposium on Agents and the Semantic Web, Arlington, VA, USA, AAAI Press (2005)
16. Sirin, E., Parsia, B.: Planning for Semantic Web Services. In Proceedings of Semantic Web Services Workshop at 3rd International Semantic Web Conference (ISWC'04) (2004)
17. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M.: SWRL: A Semantic Web Rule Language combining OWL and RuleML. (Online). (2003). http://daml.org/2003/11/swrl/
18. Li, L., Horrocks, I.: A Software Framework for Matchmaking Based on Semantic Web Technology. In: Proceedings of 12th International World Wide Web Conference (2003)
19. Jess the Rule Engine for the JAVA™ Platform. (online). http://herzberg.ca. sandia.gov/jess
20. Havey, M.: Essential Business Process Modeling. O'Rielly (2005)
21. Barros, O. H.: Business Information System Design Based on Process Patterns and Frameworks. (online). (2004). http://www.bptrends.com
22. Jena Semantic Web Framework: Jena. (online). http://jena.sourceforge.net/ index.html