International Journal of Advanced Research Methodology in Engineering & Technology, ISSN 2456 6446
Volume 1, Issue 3, May 2017

Service Identification Through Semantic Web

Subarna Panda , M.Tech C.S.E
Dept. Of Computer Science and Engineering
SRM University , Kattankulathur,Chennai,India
Email:subbarna.panda@gmail.com

S Ganesh Kumar,Asst. Professor (Sr.G)
Dept. Of Computer Science and Engineering
SRM University, Kattankulathur,Chennai, India
Email:ganesh.ks@ktr.srmuniv.ac.in

***Abstract*— Based on the studies of semi-automated composite we found that, Web services select concrete services based on the functional and/or non- functional attributes. They do not consider relationships between these attributes in the description of services or the user constraints. Thus, we propose an approach, which relates services to objects (resources) maintained by these services. The user can impose his constraints on the objects affected by the requested services. The affected object and their relationships are described in an intermediate ontology using OWL-DL and SWRL languages. Our selection strategy considers the relationships between services by looking for the dependent instances (conforming objects values) of affected objects that satisfy the user constraints and by combining the related services to get conforming composite services. The proposed selection approach of conforming composite services is implemented and integrated using Ant Colony optimization follow paths with conforming values of concrete services using SPARQL query.**

Keywords*— Concrete service; OWL; SPARQL query; Ant Colony

I. INTRODUCTION

Web services composition is a highly active studied research direction. It provides a mechanism to aggregate a multiple services into one composite service. In contrast to the traditional Web service composition based on IA planning or workflow, there are a lot of Web services offered by different providers providing the same functionality; however, these services that have different values of attributes describing functional or non-functional properties (NFP) could be gathered into a collection of Web services (category, task, type, community, abstract service) and used to select and to deter-mine the most appropriate concrete (instance) service.

The ultimate goal of the composition research is to provide a fully automatic process. Such semiautomatic Web service compositions are essentials when for example considering user constraints for each abstract service where specific Web services are not predefined. All semiautomatic approaches are based on an existing abstract composition and on instantiating this abstract composition by selecting the most appropriate concrete Web services with respect to user constraints and service properties. Selection for the purpose of the composition process depends on user constraints imposed on functional properties (FP) (for example, the user seeks a three stars hotel) or NFP (for example, the user seeks a service with high availability).It has to take care of the difference between conforming composite Web service when constraints on the FP surfaces and the optimized composite service impose constraints on NFP. Most of the solutions that calculate the optimized composite service are generally syntax-based approaches, focusing on the optimization problem and use some optimization algorithms such as Integer programming or genetic algorithms. Considering the problem of con-forming composite services, semantic-based approaches are generally used to locate and match functional attributes describing individual concrete services. However, these approaches focus on describing and locating single services. Thus, they cannot capture relationships between services that are essential information to determine their compatibility and their dependency when we select concrete services for the purpose of the composition.

The paper focuses on the objects (resources) maintained by the concrete services and their semantic relationships described in an independent ontology. Therefore, services are indexed according to these affected objects, and user constraints are imposed on the affected objects.Selection strategy considers the relationships between services when it takes into account the relationships between their related objects. To relate services to their affected objects, a semantic framework called abstract services ontology that clusters concrete services into different abstract services according to their functional characteristics is being proposed and the type of manipulated objects. Based on imposing constraints on the affected objects, the selection approach generates dynamically SPARQL queries to infer dependent instances of affected objects (conforming objects values) and related ser-vices that will be combined in order to form conforming composite services.

In this paper, a semantic selection approach of conforming composite Web services is presented. Here, contributions are threefold. First, we propose an ontology framework that allows bundling concrete services to their abstract services according to their functionality and their affected functional objects. An intermediate independent ontology has been pro-posed in order to provide a shared and common description of affected objects and their relationships. Second, showing how the user can draw a new kind of constraints, semantic constraint by considering the relationships between objects. Third, implementation of a semantic approach for the selection using semantic Web tools to locate conforming composite concrete services.

II. GLOBAL ARCHITECTURE FOR SERVICE SELECTION AND COMPOSITION

A typical architecture of our system is shown in Fig.1.Itinvolves four types of players: service providers, information agents, ontology provider, and the composer. Service providers offer the atomic concrete services. The information agents crawl the Web to collect information about concrete services from the atomic providers and index them into a meta-ontology called abstract services ontology according to the specialized functionalities (abstract services). The vocabulary shared by these abstract services called inter-mediate ontology is defined by the ontology provider. The ontology provider has also to define the offered functionalities (abstract services) and the information agent assigned to each abstract service.

Each information agent is responsible for the automatic discovery at runtime/off time of the concrete services that fulfill the corresponding abstract service functionality. Information agents gather information about concrete services such as cost, price, type, and affected objects. The obtained information is saved in the abstract services ontology and used by the composer to find the most suitable concrete services according to the intermediate ontology.

The composer has to query ontologies, to select and execute the best conforming composite concrete services. Before describing the used ontologies, we will firstly give some definitions to the used formalism.

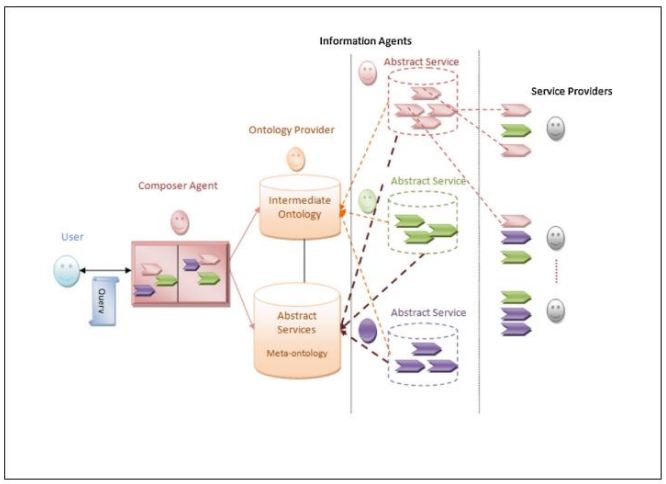

Fig. 1. Global architecture for service selection and composition

III. SYSTEM ANALYSIS

A. System Analysis

- In the existing system, all semiautomatic approaches are based on an existing abstract composition by selecting the most appropriate concrete Web services with respect to user constraints and service properties.
- A lot of existing Web services provides similar functionality.
- Most existing work of service composition and selection are based on NFP.
- Former approaches, use independent functional attributes and learning about services properties in order to avoid execution of service with backtracking mechanism.

B. Problem Definition

Have tackled only some parts of the natural features of the Web service composition problem. There is no optimization technique for selection process. No selection strategy has been used which leads to complexity on large scale. No use of querying process and intermediate ontology configuration on web service. No proper querying and optimization technique has been followed on web service.

C. Proposed System

- We propose a semantic framework called abstract services ontology that clusters concrete services into different abstract services according to their functional characteristics and the type of manipulated objects.
- Generating dynamic SPARQL queries to interdependent instances of affected objects (conforming objects values) and related services to form composite services.
- Using Ant colony optimization technique to integrate optimization and semantic selection of conforming web services.

IV. SYSTEM IMPLEMENTATION

A. Module Description

List of modules are:

1) *Query Composer*
2) *Intermediate Ontology*
3) *Abstract Services*
4) *Concrete Services*

1) Query Composer:

The composer has to query ontologies, to select and execute the best conforming composite concrete services. Before describing the used ontologies, definitions to the used formalism will be introduced.

Ontology formalism:

Web Ontology Language OWL-DL as formalism to represent ontologies will be used. OWL-DL is a decidable fragment of OWL, and based on description logics (DL), it facilitates logical inference and allows reasoning on ontologies. Usage of rule languages to express such additional primitives not provided by OWL. One of the earliest formalisms combining OWL and rules is the Semantic Web Rule Language (SWRL). The most commonly used Semantic Web query language is SPARQL, which is intended initially

to be used for resource description framework (RDF). However, this definition of SPARQL is allowed to be extended to OWL entailment.

2) Intermediate Ontology:

Intermediate ontology configuration contains only the basic concepts of services. It allows the construction of more specialized ontologies for specific application domains. Formally, the intermediate ontology (O) is defined by a TBOX and an ABOX. A T Box is a terminological component described by the tuple (C, D, DP, OP, CC, CP, POP, and Rules) where

C is a set of primitive and defined concepts like (hospital, laboratory, and physician). Defined concepts are described by using inclusion or equivalent axioms.

D is a set of data Types like (string, integer).

DP is a set of data Type property like (has Name).

OP is a set of object Property like (work With, work In); each object property has its domain and range in C.

CC is the classification relationship between classes like physician is a subclass of Doctor.

CP is the classification relationship between object Properties or data Type properties like has Name physician is a sub Property of has Name Doctor

POP are properties of object Properties like: Functional, transitive, and reflexive. Work With is a transitive Object-property.

Rules are properties, which need to be inferred. The example presented in (1) indicates that the laboratory is near to the hospital if they are located in the same country.

Hospital (?x)∧has Address(?x, ?z)

∧ laboratory (?y)∧has Address (?y, ?w)

∧located In(?z ,?t)∧located In(?w ,?t)→near(?x ,?y). (1)

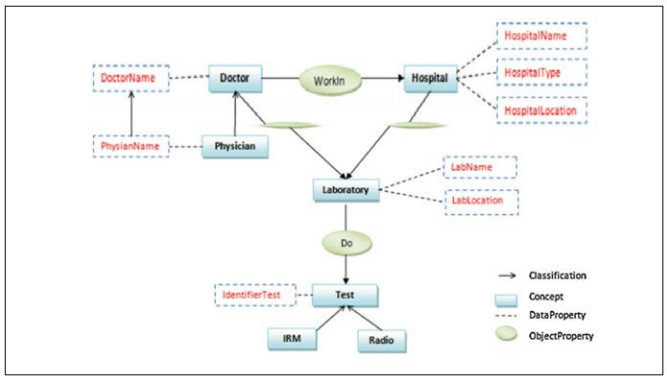

Fig. 2. A portion of our example TBOX ontology

For instance, a portion of TBOX ontology is presented in Fig.2. It shows that Physician is a subclass of the Doctor who works in Hospital. This latter interacts only with some laboratories that do some recommended tests like IRM or Radio. Hospital(h1), laboratory(l1) and work With(h1,l1)are examples of ABOX assertions associated with the TBOX describing compliant statements instances (individuals) and their relationships.

3) Abstract services:

An abstract service generally, contains the specific concepts or data needed.

An abstract description establishes the interface characteristics of the Web service without any reference to the technology used to host or enable a Web service to transmit messages. By separating this information, the integrity of the service description can be preserved regardless of what changes might occur to the underlying technology platform.

4) Concrete services:

In general, concrete service is the collection of inner classification of a specified data.

In order for a Web service to be able to execute any of its logic it needs for its abstract interface definition to be connected to some real, implemented technology. Because the execution of service application logic always involves communication, the abstract Web service interface needs to be connected to a physical transport protocol. This connection is defined in the concrete description.

B. methodologies

1) Web tools and languages :

Web Ontology Language OWL-DL as formalism to represent the ontologies. OWL-DL is a decidable fragment of OWL, and based on description logics (DL),it facilitates logical inference and allows reasoning on ontologies.

Usage rule languages to express such additional primitives not provided by OWL. One of the earliest formalisms combining OWL and rules is the Semantic Web Rule Language (SWRL). Informally, each SWRL rule has the form B→H, where Band Hare possibly empty conjunctions of atoms. However, reasoning with an arbitrary SWRL expression usually becomes undesirable. Restricting to DL-safe rules. DL-safe rules maintain the reasoning decidable by imposing constraints on the format of the rule, and this means that each variable occurring in the rule must also occur in a non-DL-atom in the body of the rule. This involves that all objects referred to in the rule have to be known explicitly.

Until now, there is no standard query language specifically for OWL and rules ontologies. The most commonly used Semantic Web query language is SPARQL, which is intended initially to be used for resource description framework (RDF). However, this definition of SPARQL is allowed to be extended to OWL entailment. A semantic specification for SPARQL compatible with OWL-DL has been defined in SPARQL using Pellet inference engine. So, to query and reason over our ontologies using OWL-DL language, we will use the Pellet inference engine that supports also DL-safe rules.

2) Abstract servies ontology:

In this section, we will show how the abstract services and concrete services can be formalized using OWL-DL. Proposing an ontology framework based on the following ideas:

1. Separation between descriptions of abstract service and that of concrete service. Concrete services are related to their abstract services by an object Property of Has Abstract Service.
2. Description of concrete services by using functional attributes and non-functional attributes. NFPs are QoS attributes. Functional attributes are those of affected objects that are described independently in the intermediate ontology. Each concrete service is related by the values of their affected object using an object Property called affects.
3. Concrete services are bundled according to their functional description (their abstract service) and the type of the affected objects by using inclusion/equivalent axioms of OWL-DL.

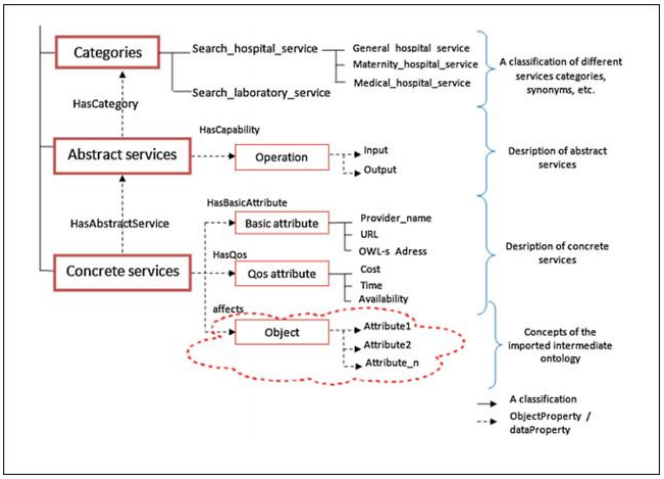

Fig. 3. Ontology framework

Therefore, Ontology framework presented in Fig.3is structured in three parts: categories, abstract services, and concrete services. In the first part, the model is based on a classification of available abstract services called categories. Abstract service described by axiom (2) represents the description of the common FP of service, which defines in general relevant concepts to specify services (independent from a real service). The functionality of a service is described by parameter types of its operations (inputs, outputs). The axiom (2) makes sure that each abstract service has at least one category and at least one functional operation.

$$\begin{aligned} AbstractService \sqsubseteq \top \; & \sqcap \exists hasCategory.Category \\ & \sqcap \forall hasCategory.Category \\ & \sqcap \exists hasCapability.Operation \\ & \sqcap \forall hasCapability.Operation. \end{aligned} \tag{2}$$

Furthermore, a service may require external conditions to be satisfied. These conditions specify restrictions that apply to the values used as inputs and outputs. Thus, the conditions used to specify restrictions (a set of values) on (input, output, precondition, and effect) model a restriction on objects. Other attributes in the concrete services layer related to the QoS and providers attributes are added to describe the specific NFP of concrete services as described in Fig.3.

One of the important characterizations of the proposed framework is that each concrete service is plugged into class of concrete services that specifies the corresponding abstract service and the type of the affected objects as presented in axiom (3).

$$\begin{aligned} & ConcreteServiceClass \equiv ConcreteService \\ & \sqcap (\exists hasAbstractService.\{Instance\_AbstractService\}) \\ & \sqcap (\exists affects.intermediate : affectedObject) \\ & \sqcap (\forall affects.Intermediate : affecetdObject). \end{aligned} \tag{3}$$

3) Service selection constraint model:

According to the abstract service ontology,extraction of affected objects of each abstract service that belongs to the composite abstract service. Furthermore, the user can also interact with the inter-face to specify his constraints on attributes describing the affected objects or on their relationships as indicate in Fig. 4. Hence, in our approach, instances of affected objects which satisfy the user constraints can be inferred over the intermediate ontology.Calling these dependent instances "conforming objects values." Using conforming values was inspired from and adapted to the affected objects.

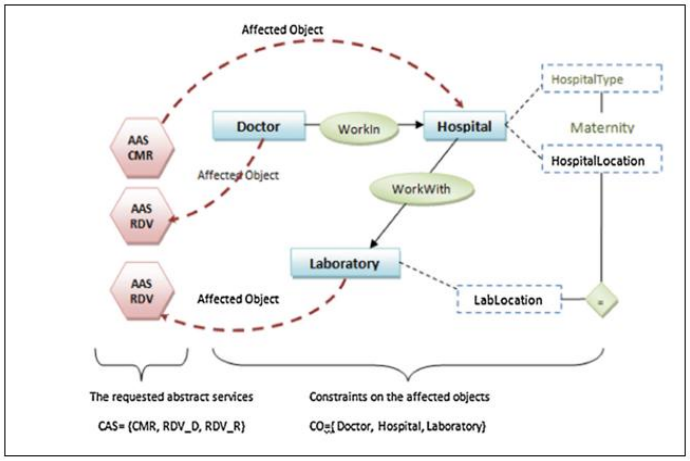

Fig. 4. User constraints on the affected objects

Only related services to the conforming objects instances can participate in the composition flow and their combination returns conforming composite ser-vices. To obtain all combinations, we would generate a Carte-sian product of candidate concrete services for all abstract services of the user request.

Definition 1(Request model):

Constraints are included in the request and imposed on objects. They can be local, global, or semantic. The local constraint (LC) restricts values of attributes of one object. For example, Hospital. type={Maternity} is a LC with parameters (Object, Attribute, value)=(Hospital, Type, "Maternity").

Global constraint involves many attributes from different objects. A global constraint is described by binary attribute comparison (BAC) that describes a comparison relationship that must exist between attributes describing concrete ser-vices. For example, hospital. Hospital Location=laboratory. Lab Location is a BAC constraint using "=" operator. A comparison operator can be either of=,<,>,≤,

≥,=,and∈. In general, a global constraint has the following parameters:object1, attribute1, object2, attribute2, operator.

Adding some semantic object relations (SOR) that describe the semantic relationships that exist between objects defined in the intermediate ontology. Imposing work In object Property (See Fig.4) for example, between Doctor and Hospital concepts can reduce significantly the number of affected objects values. Therefore, semantic constraint has parameters of object1, object2, semantic relationships.

So request template (RT) will be

RT=(CAS,CO,RM)and

RM=LC+BAC+SO R. (4)

Where,

CAS: composite abstract service.

CO: composite object of CAS, which contains the set of the affected objects of its atomic abstract services.

Definition 2(Conforming objects values):

Conforming objects values for a given RM are the set of tuples of instances of a composite object (CO) where all constraints of a given request model (RM) are satisfied.

$$COI(RM, CO) = \{(i_{11}, i_{12}, \ldots i_{1n}), \ldots (i_{m1}, i_{m2}, \ldots i_{mn})\}$$

$$Where$$

$$(i_{j1}, i_{j2}, \ldots i_{jn})\ satisfy\ \ RM,\ \forall j \in 1 \ldots m. \quad (5)$$

where

i_{j1} is an instance of the object O_1, $\forall j \in 1 \ldots m$

i_{j2} is an instance of the object O_2, $\forall j \in 1 \ldots m$

i_{jn} is an instance of the object O_n, $\forall j \in 1 \ldots m$

$$CO = \{O_1, O_2, \ldots O_n\}$$

Definition 3(The selection constraint model):

The naive selection of conforming concrete services for a composite abstract service consists of finding all the combination of concrete services that satisfy user constraints as indicated in(6a).

$$Conforming(CAS, RM) = \{csc_1, \ldots csc_m\} \quad (6a)$$

$$where$$

$$\{\{for\ each\ aas_j \in CAS,$$

$$\{Discovery(aas_j) = ACS_j\} \quad (6b)$$

$$V = \left\{ calculate \prod_{j=1}^{|CAS|} ACS_j \right\}$$

$$\} \quad (6c)$$

$$csc_i = x \in V\ where\ RM\ in\ x\ is\_valid() \quad (6d)$$

$$\}$$

Where,

aasj: atomic abstract service.

csci: composite concrete service.

AC Sj: set of atomic concrete services.

CAS: composite abstract service.

is_valid(): is a function that verifies satisfaction of con-straints on a csc.

Table 1 List of concrete services with their affected objects

Abstract service	Concrete services	Affected objects
AS_MR	cs1	h1, h2
	cs2	h2
AS_RDVD	cs3	d1, d2
	cs4	d3
AS_RDVL	cs5	l1, l3
	cs6	l2
	cs7	l1

Table 2 List of conforming objects values:COI(RM1,CO)

Doctor	Hospital	Laboratory
d1	h1	l1
d3	h1	l1

Table 3 List of related concrete services for each object value

Doctor	Hospital	Laboratory
{cs3}	{cs1}	{cs5,cs7}
{cs4}	{cs1}	{cs5,cs7}

Conforming instances of affected objects can be identified by querying independently the intermediate ontology, which describes affected objects and their relationships. Based on the abstract services ontology where we index concrete ser-vices according to their affected objects, we can for each instance (x) of a given object find the related concretes services(discover y(aasj, x)) by querying the abstract services ontology.

Let's consider the abstract services ontology for the above scenario. Affected objects for each concrete service are listed in Table1, showing how they are indexed by the object Property "affects."According to the following RM imposed by the user:

RM1={ Hospital. type=maternity, hospital. Hospital Location=laboratory. Lab Location, Doctor work_in Hospital, laboratory work_with Hospital} and with a CO={doctor, hospital, laboratory}, suppose that querying the intermediate ontology by the RM1 generates only values presented in Table2. Each line of this table represents conforming objects tuple from the set COI(RM1, CO).

According toTable1, the valuel1forLaboratoryobject is related to the two concrete services {sc5, cs7}, to be returned by the function Discovery (RDV_L,l1). So for each value of the table 2, a list of related concrete services according to the abstract services ontology will be generated as indicated in Table3(8c).

Table 4 List of conforming composite concrete services

AAS CMR	AAS RDV_D	AAS RDV_L
cs1	cs3	cs5
cs1	cs3	cs7
cs1	cs4	cs5
cs1	cs4	cs7

Each list of these related concrete services is considered as a conforming candidate concrete services for one atomic abstract service. The Cartesian product of these lists is done (8d) for each line of table Table3 to obtain the final set of conforming CSC(8e): {cs3.cs1.cs5, cs3.cs1.cs7, cs4.cs1.cs5,cs4.cs1.cs7}, as indicated in Table4. Each line in this tables a conforming composite concrete service.

On the other side, if we consider the naive approach, we would generate 2*2*3 composite concrete services before verifying user constraints instead of 4 composite concrete services located in the proposed approach. In the next section, we will show how to realize our proposed selection approach, based on the abstract services ontology and the intermediate ontology.

Service selection constraints model using SPARQL queries:

The approach is composed of three main phases: (1) preprocessing, (2) identifying conforming objects values, and (3) identifying con-forming composite services. In the first phase, XML files to structure RM of the user: list of the requested abstract services, their affected objects, and the user's constraints. The second phase determines instances of the affected objects that are conforming to users constraints as shown in (8b). To determine these instances, query the intermediate ontology using SPARQL language. While user constraints on the affected objects can be structured and rewritten in a SPARQL query called SPARQL query1, the result returned after its execution at runtime is a table containing instances of dependant affected objects, each line in this table is a tuple of the conforming affected objects. The third phase locates services that are related to each value returned by the last SPARQL query1 (8c). According to the abstract ser-vices ontology where we index services to their affected objects, we need to query this ontology by another SPARQL query said SPARQL query2 to return related services at each instance of the affected objects. Cartesian product of related services for each tuple of COI will be doing to generate and return set of all composite conforming services (8a).

Preprocessing:

In the approach, the semi-automatic composition process starts when the user draws the composition flow using a set of tools in the interface for modeling the composition process as a combination of abstract services (or categories) and control flow. Based on this module, a first XML file will be generated containing a list of the different abstract services. We generate a second XML file by adding automatically to the first one the

affected objects to each abstract service according to the abstract service ontology, by considering axioms define in Sect.3and the Objects Property of "affects."

```
<abstractService>
   <name>...</name>
   <affectedObject>...</affectedObject>
</abstractService>
```

Based on these affected objects and their properties, the user can impose his constraints. Therefore, we add to the second XML file all LC imposed on attributes of each object, all the relationships between the different attributes (BAC) of affected objects and the semantic relationships (SOR) according to the intermediate ontology.

```
<localConstraint>
  <Object: attribute_i>...
      </Object: attribute_i>
  <value>...</value>
</localConstraint>

<globalConstraint>
  <Object1: attribute_i>...
      </Object1: attribute_i >
  <Object2: attribute_j >...
      </Object2: attribute_j >
  <binaryOperator>...</binaryOperator>
</globalConstraint>

<semanticConstraint>
  <Object1>...</Object1>
  <Object2>...</Object2>
  <ObjectProperty_k>...
     </ObjectProperty_k >
</semanticConstraint>
```

Compared to the intermediate ontology components, we find that LC is a restriction on a Data Property, BAC is also a restriction on two Data Property that must be compared, and SOR is an Object Property that must exist between two concepts. Therefore, extraction from the last XML file and will generate automatically a SPARQL query containing user restrictions.

Conforming object values determination:

There will be one SPARQL query to generate for all the different types of constraints. Based on the second XML file2 generated in the previous phase, a generator will rewrite constraints in SPARQL language as:

```
1  PREFIX onto:<http://www.owl-
           ontologies.com/Intermediate
           Ontology.owl#>
2  SELECT ?object1 ?object2...?object_n
3  WHERE {
4     ?object1 rdf:type onto:Object1.
5     ?object2 rdf:type onto:
         Object2.....
6     ?object_n rdf:type onto:Object_n.

7     ?object_k onto:ObjectProperty_j
           ?object_k'.
8     ?object_i onto:DataProperty_1
         ''value_1''.
9     ?object_i' onto: DataProperty_1'
           ?w.
10    ?object_i'' onto: DataProperty_1''
         ?r.
11 FILTER(?w binaryOperator ?r ).}
```

Where,

2: object1, object2..., object_n are variables that correspond to the affected objects. After the execution of this query, these variables return instances of the affected objects (4–6), which satisfy the list of the constraints (7–11).

4–6: Object1, Object2,... Object_n are the list of all affected objects listed in XML file2.

7:ObjectProperty_j is the semantic relationships imposed by the user in the SOR between the affected objects (object_kand object_k').

8: Data Property_l is an attribute of the affected object described in the local constraints. value_l is its restriction value imposed in LC of the XML file 2.

9–10: Data Property_l' and Data Property_l" are attributes of two objects (object_i' and object_i") to be compared according to the BAC, filtered in11. FILTER construction of SPARQL query to compare two attributes will be castoff.

The role of this SPARQL query1 is to specify constraints on the affected objects and return after execution only conforming objects values. The execution of this query over the intermediate ontology will use Description logic and rules reasoning to infer direct or indirect instances values of object1, object2,... object n that satisfy SPARQL query constraints listed in WHERE part of the query.

An example of SPARQL query1 for the previous RM1 is

```
PREFIX onto:<http://www.owl-ontologies.
                com/Intermediate
                Ontology.owl#>
SELECT ?doctor, ?hospital, ?laboratory
WHERE {
```

```
        ?doctor rdf:type onto:Doctor.
        ?hospital rdf:type onto:Hospital.
        ?laboratory rdf:type onto:
             Laboratory.

        ?doctor onto:workIn ?hospital.
        ?hospital onto:workWith
            ?laboratory.
        ?hospital onto:hasType \
          ''Maternity''.
        ?hospital onto:hasAdress ?w.
        ?laboratory onto:hasAdress ?r.
FILTER ( ?w = ?r).}
```

Conforming composite concrete services determination:

The main idea to discover lists of candidate concrete services (8c) is to generate for each value returned by the above step the related concrete services according to the abstract service ontology. For each value, it will generate a simple SPARQLquery2 (line 6 of Algorithm 1). The result returned after the execution of this query is a table with one column of concrete services saved in a list (line 7). The SPARQL query2 has two parameters [type of atomic abstract service (aasj) and the specified value (COI)] as indicated in (line 6). Its general form is

```
PREFIX onto1: http://www.owl-ontologies
                com/ AbstractService
                Ontology.owl#
SELECT ?x
WHERE {?x onto:hasAbstractService
           Abstract_S[j].?x onto:
           affects onto1: A[i,j].}
```

As an example for the value of "h1" of the Hospital object, we generate the following query:

```
PREFIX onto1: http://www.owl-ontologies.
                com/AbstractService
                Ontology.owl#
SELECT ?x
WHERE {?x onto:hasAbstractService onto:
        CMR.?x onto:affects onto1:h1.}
```

To avoid generating the same query more than one and because some values could be occurred in different lines of the table returned by SAPRQL query1, as indicated in Table2, create another list (Treated Value) (line 1) to verify if for a given value, we have not previously (line 5) generate the corresponding query as described in lines 5 and10.

As described in Table2, the table result of SPARQL query1 indicates that each line (tuple) in this table is a conforming value for the different objects, so we initialize vector lists (lines 4 and 12) and generate a Cartesian product (lines 13–21) for each tuple before moving to the next tuple (lines 3 and 22). Line 23 displays conforming concrete services. Moreover, algorithm1 use in lines 13–21 product of lists to generate Cartesian product for a non-predefined abstract service number.

Algorithm 1: Algorithm of Locating conforming composite concrete services

```
   Input: table A[1..n, 1..m]: result of the SPARQL Query1
   Input: Abstract_Services[1..m]: list of the abstract services
   Result: SC: list of the conforming concrete services
1  [TreatedValue]= null;
2  A = result of SPARQL Query1;
3  for i ← 1 to n do
4  |  for j ← 1 to m do
5  |  |  if A[i, j] ∉ [TreatedValue] then
6  |  |  |  SPARQL_Query2 =
   |  |  |  Generate_Query(Abstract_Service(j), A[i, j]);
7  |  |  |  [Sj] = execute(SPARQL_Query2);
8  |  |  |  Add A[i,j] to [TreatedValue] ;
9  |  |  else
10 |  |  |  [Sj]= Return [Sj] of A[i,j] from [TreatedValue];
11 |  |  end
12 |  end
13 |  //Generate [S1]x[S2]x[Sm]: phase(8.d)
14 |  for each service sa in [S1] do
15 |  |  for each service sb in [S2] do
16 |  |  |  ...;
17 |  |  |  for each service sm in [Sm] do
18 |  |  |  |  SC.add(sa, sb,...,sm);
19 |  |  |  end
20 |  |  end
21 |  end
22 end
23 Display (SC)
```

Ant Colony Optimization:

The ant colony algorithm is an algorithm for finding optimal paths that is based on the behavior of ants searching for food.

At first, the ants wander randomly. When an ant finds a source of food, it walks back to the colony leaving "markers" (pheromones) that show the path has food. When other ants come across the markers, they are likely to follow the path with a certain probability. If they do, they then populate the path with their own markers as they bring the food back. As more ants find the path, it gets stronger until there are a couple streams of ants traveling to various food sources near the colony.

Because the ants drop pheromones every time they bring food, shorter paths are more likely to be stronger, hence optimizing the "solution." In the meantime, some ants are still randomly scouting for closer food sources. A similar approach can be used find near-optimal solution to the traveling salesman problem.

Once the food source is depleted, the route is no longer populated with pheromones and slowly decays.

Pseudo-code and formula:

```
procedure ACO_Meta Heuristic
while (not_termination)
    generateSolutions()
    daemonActions()
    pheromoneUpdate()
  end while
end procedure
```

Because the ant-colony works on a very dynamic system, the ant colony algorithm works very well in graphs with changing topologies. Examples of such systems include computer networks, and artificial intelligence simulations of workers.

V. CONCLUSION

In this paper, we have proposed a semantic approach to select conforming composite Web services based on imposing constraints on the affected objects of the requested services. Our selection for the purpose of the composition locates dependent services by considering the relationships between affected objects described in an intermediate ontology. Our selection implementation generates dynamically SPARQL queries to obtain conforming objects values and related concrete services.

VI. FUTURE ENHANCEMENT

The main characteristics of our approach are the fact that we neglect non-conforming Values and we generate a polynomial number of SPARQL queries to reduce execution time. Imposing constraints on the affected objects reduces significantly the number of conforming objects values and so forth the number of generated composite services.

References

[1] B. Franc¸ois, M.-A. Nolin, N. Tourigny, P. Rigault, and J. Morissette, "Bio2RDF: towards a mashup to build bioinformatics knowledge systems," J. biomedical informatics , vol. 41, no. 5, pp. 706–716, 2008.

[2] EHealth Information Platforms (EHIP). [Online]. Available: http:// distrinet.cs.kuleuven.be/ research/projects/EHIP, Dec. 2013

[3] Axiomatics Language for Authorization (ALFA). [Online]. Available: http://www.axiomatics.com/solutions/products/ authorization-for-applications/developer-tools-and-apis/192- axiomatics-language-for-authorization-alfa.html

[4] Sun's XACML Implementation. [Online]. Available: http:// sunxacml.sourceforge.net/, 2003.

[5] WSO2 Balana Implementation. [Online]. Available: https:// github.com/wso2/balana, 2013.

[6] D. Agrawal and C. C. Aggarwal, "On the design and quantification of privacy preserving data mining algorithms," in Proc. SIGMOD-SIGACT-SIGART Symp. Principles Database Syst., 2001, pp. 247–255.

[7] R. Agrawal and C. Johnson, "Securing electronic health records without impeding the flow of information," Int. J. Med. Inf., vol. 76, pp. 471–479, 2007.

[8] C. P. Antonopoulos, V. Kapsalis, and L. Hadellis, "Optimal scheduling of smart homes' appliances for the minimization of energy cost under dynamic pricing," presented at the 17th Int. Conf. Emerging Technol. Factory Autom., Krakow, Poland, Sep. 17–21, 2012.

[9] M. Barhamgi, D. Benslimane, C. Ghedira, and A. L. Gancarski, "Privacy-preserving data mashup," in Proc. Int. Conf. Adv. Inf. Netw. Appl., 2011, pp. 467–474.

[10] R. Bhatti, E. Bertino, and A. Ghafoor, "A trust-based contextaware access control model for web-services," in Proc. Int. Conf. Web Services, 2004, pp. 184–191.J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.

[11] G. Brown, A. Pocock, M.-J. Zhao, and M. Lujan, "Conditional likelihood maximisation: A unifying framework for information theoretic feature selection," J. Mach. Learn. Res., vol. 13, pp. 27–66, 2012.

[12] A. D. Brucker and H. Petritsch, "Idea: Efficient evaluation of access control constraints," in Proc. 2nd Int. Conf. Eng. Secure Softw. Syst., 2010, pp. 157–165.

[13] M. Decat, B. Lagaisse, and W. Joosen, "Middleware for efficient and confidentiality-aware federation of access control policies," J. Internet Services Appl., vol. 5, no. 1, pp. 1–15, 2014.

[14] A. Erradi, P. Maheshwari, and V. Tosic, "Policy-driven middleware for self-adaptation of web services compositions," in Proc. ACM/IFIP/USENIX Int. Conf. Middleware, 2006, pp. 62–80.

[15] R. Ferrini and E. Bertino, "Supporting RBAC with XACML+ OWL," in Proc. 14th ACM Symp. Access Control Models Technol., 2009, pp. 145–154.

[16] T. Grandison, S. R. Ganta, U. Braun, and J. Kaufman, "Protecting privacy while sharing medical data between regional healthcare entities," Stud. Health Technol. Inf., vol. 129, pp. 483–487, 2007. AMMAR ET AL.: XACML POLICY EVALUATION WITH DYNAMIC CONTEXT HANDLING 2587

[17] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: An update," ACM SIGKDD Explorations Newslett., vol. 11, pp. 10–18, 2009.

[18] B. Kabbani, R. Laborde, F. Barrere, and A. Benzekri, "Specification and enforcement of dynamic authorization policies oriented by situations," presented at the 6th Int. Conf. New Technol., Mobility Security, Dubai, United Arab Emirates, Mar. 30–Apr. 2, 2014.

[19] S. Kasthuri and T. Meyyappan, "Detection of sensitive items in market basket database using association rule mining for privacy preserving," in Proc. Int. Conf. Pattern Recog., Inf. Med. Eng., 2013, pp. 200–203.

[20] R. Laborde, B. Kabbani, F. Barrere, and A. Benzekri, "An adaptive XACMLv3 policy enforcement point," in Proc. IEEE 28th Int. Comput. Softw. Appl. Conf. Workshops, 2014, pp. 620–625.

[21] K. LeFevre, R. Agrawal, V. Ercegovac, R. Ramakrishnan, Y. Xu, and D. DeWitt, "Limiting disclosure in hippocratic databases," in Proc. 13th Int. Conf. Very Large Data Bases, 2004, pp. 108–119.

[22] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan, "Mondrian multidimensional k-anonymity," in Proc. 22nd Int. Conf. Data Eng., 2006, pp. 25.

[23] N. Li, T. Li, and S. Venkatasubramanian, "t-closeness: Privacy beyond k-anonymity and l-diversity," in Proc. IEEE 23rd Int. Conf. Data Eng., 2007, pp. 106–115.

[24] A. X. Liu, F. Chen, J. Hwang, and T. Xie, "Designing fast and scalable xacml policy evaluation engines," IEEE Trans. Comput., vol. 60, no. 12, pp. 1802–1817, Dec. 2011.

[25] A. Machanavajjhala and J. Gehrke, "On the efficiency of checking perfect privacy," in Proc. 25th ACM SIGMOD-SIGACT-SIGART Symp. Principles Database Syst., 2006, pp. 163–172.

Authors :

SUBARNA PANDA
M.Tech- C.S.E
SRM University
Kattankulathur, Chenna

S GANESH KUMAR
Asst. Prof (Sr.G)
SRM University,
Kattankulathur, Chennai