I.J. Information Technology and Computer Science, **2015, 12, 30-39**
Published Online November 2015 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2015.12.04

SQUIREL: Semantic Querying Interlinked OWL-S traveling Process Models

Afaf Merazi
EEDIS Laboratory, Computer Science Department. Djillali LIABES University, Sidi Bel Abbes, Algeria
Email: afaf.merazi@univ-sba.dz

Mimoun Malki
High School of Computer Science of Sidi Bel Abbes (ESI - Sidi Bel Abbes) Algeria
Email: malki@univ-sba.dz

Abstract—With the advent of new forms of information and communication technologies, the consumer needs to combine and customize different travel components as a complete travel package, namely: Dynamic Packaging Technology. Nevertheless, disparate tourist offers and services make it difficult for consumer to use them effectively. Therefore, our paper presents an intelligent querying framework of OWL-S travel services, called SQUIREL composition engine. It uses Semantic Web Services (SWSs) technologies combined with the useful of Linked e-tourism Data concept to fulfill the preferences and constraints of the e-tourist any time. This purpose supports SWSs pre-selection through the valuation of the rewritten SPARQL consumer query at runtime that manages dynamic service dependencies extracted from Linked e-tourism Data and returns the SWSs endpoint. Then, SQUIREL catches this endpoint and makes the necessary optimizations to refine it to its relevant atomic processes needed to be composed using matrix computation. However, the experimental results indicate that this method owns both lower computation cost and higher success ratio of fine-grained discovery-based atomic processes composition.

Index Terms—Linked e-tourism data, Dynamic Packaging Technology, Automatic discovery-based composition, Semantic Web Service, OWL-S Traveling Process Model, Semantic Query (P)SPARQL.

I. Introduction

Nowadays, e-tourism supports more innovative and sophisticated tasks encouraging the consumer to seek for more personalized and customized tourism offers with a characterized need of efficiency and sense of control anytime and anywhere, called Dynamic Packaging (DP) [1]. Cardoso[2,3] is the pioneer for developing a valid DP product involved in the SEED (SEmantic E-tourism Dynamic packaging) project and described as a multilayer framework that integrates Semantic Web Services as semantic mediators ready to be composed anytime, Bilbao [4] presents also a semantic e-business platform that composes the appropriate processes given up the consumer's desires and restrictions. Unfortunately combining and booking disparate components is still a time consuming and a challenging task, due to the continuous overloaded travel's information[1] and booking platforms [2]. In order to provide an intelligent and proactive access to relevant high quality online travel information and services, the DP can particularly benefit from: (1) Linked e-Tourism Data[3] [5,6,7] that integrates business offers across different data sources and (2) Semantic Web services (SWSs) [8] technologies to book multiple travel components as a complete travel package. Therefore, this paper describes a tailored framework for consumers that group the appropriate travel atomic processes (indivisible operation defined in OWL-S process model) they are interested in, Called SQUIREL. Our proposal employs the notion of Linked e-tourism Data field that achieves the efficiency of the automatic OWL-S [8] discovery-based composition by supplying the necessary schema-based alignment [9] to rewrite the consumer query for pre-selecting the appropriate travel process models. However, the pre-selection result needs to be refined to its relevant sub-set atomic processes in order to find the final and optimal composition solution. Therefore, applying a local optimization technique is going to generate a set of sub-workflows represented as a sub-matrices needed to be merged in new one on which we apply a matrix computation technique that returns the new composite service. The proposed approach conducts fast service composition and proved to be very effective and efficient in determining the optimal atomic processes composition plan.

The rest of the paper is organized as follows: background is presented in Section 2; Section 3 describes a motivating travel scenario showing the meaningful of fine-grained discovery-based service composition, an architecture framework is detailed in Section 4. Section 5 presents the prototype implementation and the evaluation of our approach. Finally, related work is discussed in Section 6 followed by concluding remarks and future work.

[1]Consumer can get information on routes, timetables, seat availabilities, book rental cars and restaurants etc.
[2] Such as online travel portals, like Expedia.com, Travelocity.com, Orbitz.com and Kayak.com.
[3] Available at: http://datahub.io/dataset?tags=tourism.

II. BACKGROUND

The *Semantic Web* [10] is defined as a web of data, comprised of resources machine readable and connected with links where a resource (the subject (s)) is linked to another one (the object (o)) through an arc labeled with a third one (the predicate (p)), using Resource Description Framework (RDF) [11] language that represents meta-data as an RDF Graph (a set of RDF triples). An RDF triple is formalized as a tuple $< s, p, o > \in (U \cup B \cup L) \times (U \cup B) \times (U \cup B \cup L)$ where U denotes the set of URIrefs, B the set of blanks and L the set of literals, are three pair wise disjoint infinite sets but their union forms the RDF terminology T . The ontology (Web Ontology Language (OWL)) [12] defines a common vocabulary for representing knowledge about a domain and stores several concepts organized on different hierarchic levels and a set of relationships (object/data-type properties) that necessarily hold among those vocabulary terms and instances.

SPARQL [13, 14] is the most popular RDF query language, defined on top of basic graph patterns as RDF graphs with variables (from a set V), a set of RDF triple patterns, where each triple pattern is a tuple $< s, p, o > \in (T \cup V) \times (U \cup V) \times (T \cup V)$. Combining RDF triple patterns is possible using SPARQL operators such as AND- Grouping, Union-patterns, Nesting, Optional parts and Filtering Query Modifiers...A SELECT SPARQL query is expressed using a form resembling the SQL SELECT query. Answering to an RDF query is seen as a graph homomorphism from SPARQL graph pattern into an RDF Graph and defined as a partial mapping function from V to RDF terms ∈ T; $\mu : V \rightarrow T$. In order to reduce the search space over RDF graphs, Alkhateeb [15] define an extension of SPARQL query language with the use of a set of regular expression patterns over predicates characterizing the traversed paths of arbitrary length in a query called Path SPARQL (PSPARQL). For instance, the following query returns all cities reachable from the city Sydney by a sequence of trains and planes only:

```
SELECT ?city
WHERE { dbpedia-owl:Sydney (tio:train | tio:bus)+ ?city }}
```

Semantic Web Services (SWSs) are software components that are incorporated and reused into distinct dis-tributed applications without the concern of how the service was implemented. We focus on the most promising SWSs language, the OWL-S (Web Ontology Language for Web Services) [8], a kind of OWL ontology specification that formal-izes semantically service capabilities by providing: (1) the service profile that presents the Inputs, Outputs, Preconditions and Effects (IOPEs) and service category, (2) the grounding provides the needed details about transport protocols, (3) the process model (PM) presents the behavior of the service as a process, either atomic or composite which receives and sends a single message or retains/changes state through a sequence of messages, where:

- An **atomic process (AP)** is a directly invocable and executed entity that describes: (a) IOs parameters ex-pressed as a subclass of the parameter class in OWL-S and (b) PEs modeled as logical formulas or expressions.
- The **Composite processes** provides a concrete and dynamic semantic description of the logical execution order of the finite set of sub-processes that are connected to each other using OWL-S control constructs: Sequence, Split, Split+ Join, Unordered, Choice, If-Then-Else, Iterate and Repeat-Until/Repeat- While. Their IOPEs are described with concepts formally defined by means of ontologies.

III. RESEARCH MOTIVATING EXAMPLES

In order to better illustrate our approach, we consider a travel scenario where a consumer wants to book a hotel room where the conference event takes place and want to check the weather condition of this city'event.

Given four OWL-S process models that are depicted in Figure 1, where each OWL-S process model describes the internal behavior of its travel atomic processes in accordance with their semantic functionalities cross different travel domain ontologies: The first service manages an event registration with payment process based four atomic processes; The second one allows the consumer to book a star rating accommodation based two Atomic processes; The third one allows to choose or to book a room hotel relying on three atomic processes. Finally, the last one checks the weather condition of a city name or of geographical coordinates. However, these four services are closely dependent on the others due to the fact that the outputs of EventReservationService produces the inputs of the three remaining, since all process models need to work together on strategy so as to satisfy the requirement of consumer.

Fig. 1. OWL-S Process Models Examples

IV. SQUIREL FRAMEWORK

We propose a SPARQL-driven approach (L0) (see figure 2), a uniform way for: (1) searching information needed from Linked e-Tourism Data [16] (through the second Layer (L1)), and/or (2) composing automatically different OWL-S travel services (through the third layer (L2)).

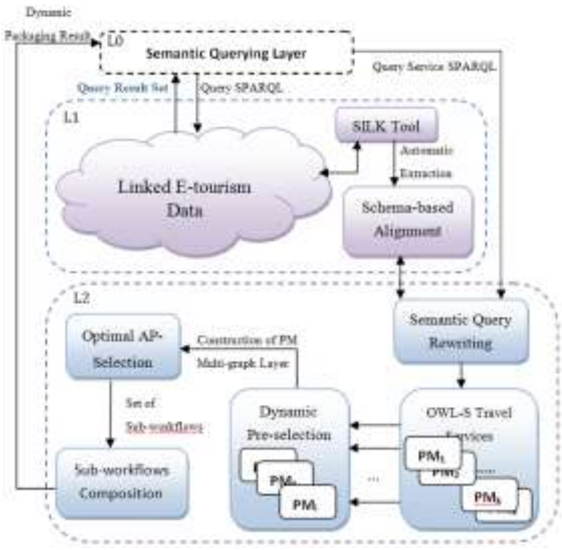

Fig. 2. A proposed intelligence traveling architecture

4.1. Semantic Querying Layer (L0)

While Linked e-Tourism Data offer for consumers a way to search and retrieve travel information using SPARQL semantic query language [16], Garcia [17] offer an efficiency approach enabling SWSs to be discovered and used by other ones using two different SPARQL concrete queries that improves the execution time. The first one, the **Qall filter** finds a reusable SWS that fulfills the whole set of related terms described by the consumer query but the second one, the **Qsome filter** offer more flexibility and returns a suitable composition of SWSs that contain at least one of the terms referred by the consumer query and so, can be composed to fulfill the requirements. The Input/output of a process model (PM), is a message received/delivered, corresponding to the <process:hasInput> /<process:hasOutput> properties in OWL-S codes. The formalization of OWL-S process model functionalities (I/O parameters) in RDF triple using Turtle-based syntax is in the form:

<PM, pr:hasInput,I_i>…<PM,pr:hasOutput,O_j>. So, for answering complex consumer request, single PM is rarely used and so, the construction of a new composite service by integrating and reusing the existing ones is required using the **Qsome filter**.

Definition1. As input of SQUIREL, a 3-tuple DPS = (T, RQ, PMs) that represent a discovery process where T is the union of several travel ontologies involved in LED, T = $\{T_1,T_2,\dots,T_m\}$ a consumer SPARQL query $R_Q = (I_Q,O_Q)$ that specifies a set of ontological concepts describing the provided inputs I_Q and requested outputs $O_Q (I_Q, O_Q \subseteq T_Q \subseteq T)$ and PMs a set of travel SWSs belonging to NAICS Travel services category (5615).

Definition2. A PM is defined as a 4-tuple (AP, E, I_{PM}, O_{PM}) where the atomic processes (APs) are the set of indivisible operations, I_{PM} is a set of inputs parameters required to invoke PM, O_{PM} is the set of outputs parameters returned by PM after the execution of PM ($I_{PM}, O_{PM} \subseteq T_{PM} \subseteq T$) And E contains the control flow relation between processes. Both of R_Q and PMs are described at a semantic level using well-defined ontological concepts defined in T.

Given the example defined in section 3, we formulate R_Q using the **Qsome filter** by employing UNION or FILTER SPARQL operator that delivers acc:Hotel or tio:Event ... and provides s:eventReservation or s:LodgingReservation, as follows:

```
SELECT ?s WHERE {
?s pr:hasInput ?I;
?s pr:hasOutput ?O;
Filter (((?I=acc:Hotel) || (?I=tio:Event) ||...)
      && ((?O=s:eventReservation) ||(?O=s:LodgingReservation )...)) }
```

4.2. Interlinked E-Tourism Data Layer (L1)

Linked data [18,19] is defined as a vast, distributed data space that use many different vocabularies in different data formats[4] build on a simple set of standards[5] where the entities are interlinked for creating a vast collection of data graph[6] that spans data sources and enables the discovery of new data sources. However, some approaches deals with linked data cloud in e-tourism domain, there may be mentioned: TourMISLOD [5] and OpeNER [6,7] (Open Polarity Enhanced Name Entity Recognition). Our framework uses an existing Linked e-tourism Data [7] allowing us to find the binary semantic relationships between two concepts (a computed pair-wise similarity): (1) equivalent concept degree (Exact Match (≡), a symmetric predicate) or (2) sub-concept degree (Plugin Match($\hat{o}$)) defined as below:

Definition3. [Generalized Concept alignment relationship] Given two concepts A and B, defined in two ontologies O_1 and O_2 such as $A \in O_1 and\ B \in O_2$ .We say that $A \rhd B$ if $(A \equiv B) \vee (A \prec B)$ where:

1. $A \equiv B$, A is semantically equivalent to B, and
2. $A \prec B$, A is a sub-Concept of a B iff:
   a. ($A \equiv C$) and ($C\ \hat{o}\ B$) where ($C \in O_2$)
   b. ($A\ \hat{o}\ D$) and ($D \equiv B$) where ($D \in O_1$)

There are several approaches [9, 21] that addresses the interlinking discover process that takes two datasets as input and produces automatically a collection of alignment between concepts of the two datasets as output across ontologies called Schema-based alignment using instance alignment techniques, the most widely used is SILK [20] that handle large volumes of data and obtain good results with high precision. There may be mentioned some concept mappings related to the example that are useful for the discovery process:

[4] So there are many different ways to represent the same information
[5] RDF, URIs, HTTP
[6] Linked Datasets (i.e., with Dereferenceable URIs) available as RDF Dumps http://www.w3.org/wiki/DataSetRDFDumps

```
vcard:Person       owl:equivalentClass   foaf:Person;
dbpedia:event      owl:equivalentClass      tio:Event;
s:Hotel             owl:equivalentClass     Acc:Hotel;
dbpedia:Hotel      owl:equivalentClass        s:Hotel;
dbpedia:startDate owl:equivalentClass   s:startDate;
dbpedia:endDate    owl:equivalentClass      s:endDate;
dbpedia:City       owl:equivalentClass         s:City;
dbpedia:Country    owl:equivalentClass      s:Country;
dbpedia:City       owl:equivalentClass      lgd:City;
dbpedia:Country    owl:equivalentClass lgd:Country;
acc:Hotel          rdfs:subclassof acc:accommodation;
```

4.3. *Interlinked Transactional Services (L2)*

This layer proposes a flexible service composition that invokes services on-demand and at runtime due to the changing environment.

4.3.1. *Basic Representation of Travel Services*

In order to store the knowledge derived from PMs behavior, we exploit the use of graph theory [22] that benefit from the use of matrix theory due to the fast access to its nodes and specially the adjacency matrix A = (V, E) that is an N-vertex directed graph where V is a finite set of vertices and E is a set of directed edges. Element A_{ij} = 1 if and only if the edge (I, j) ∈ A. All other elements are zero. A row of A lists the nodes at the tip of the outgoing edges while a column of A lists the nodes at the tail of the incoming edges. Based on this, the adjacency PM matrix (APM) is an N-square binary Boolean matrix that represents the dependencies between the AP by analyzing the complete behavior of the service, APM $=[AP_{ij}]_{NxN}$ (N denotes the number of atomic processes). The composite process determines the inter-PM dependency between APs such as if a sequence control construct edge connects two vertices from AP_i to AP_j (seq(AP_i,AP_j) is in E(PM)), APM_{ij} = 1. If there is no such edge in E(PM), APM_{ij} = 0. The matrix have zeros diagonal with no self loop that implies all services are independent on itself and form an acyclic dependency on it. Therefore, parsing the nested structure of OWL-S control structures in a top-down manner allow us to rewrite logically each complex structured process (choice, if-then-else, split+join and split) [23] to a simpler form in terms of sequence control construct as below:

- $Seq(S_1, S_2) \rightarrow PM[S_1, S_2] = 1$.
- $Seq(S_1,\alpha(S_2,S_3)) \rightarrow$ From S_1 we can go to S_2 and/or S_3, we need to add two edges to APM : $Seq(S_1,S_2) \wedge Seq(S_1, S_3) \rightarrow PM[S_1,S_2]=1,\ PM[S_1, S_3] =1$ where α = Split | IfthenElse .
- $Seq(\alpha\ (S_1, S_2), S_3)) \rightarrow$ From S_1 we can go to S_2 and/or S_3, we need to add two vertices to APM : $Seq(S_1, S_3) \wedge Seq(S_2, S_3) \rightarrow PM[S_1, S_3] = 1$, $PM[S_2, S_3] = 1$ Where α = Split+join | Choice .

Let us consider now the inter-PM dependency of EventReservationService, presented in Section 3, it contains four atomic processes and consists of three sequencing edges that connects three sub-processes SelectEvent (E1), SignInEvent (E2) and a Choice process between BankTransferPayement (E3) and creditcardpayement (E4). The service illustrates a sequence edge from E1 to E2 and from E2 to E3 or E4. In conclusion, E(PM)= { Seq(E1,E2), Seq(E2,E3), Seq(E2,E4)}. Based on the description set out above, we present the adjacency EventReservationService matrix depicted in Figure3:

	E_1	E_2	E_3	E_4
E_1	0	1	0	0
E_2	0	0	1	1
E_3	0	0	0	0
E_4	0	0	0	0

Fig. 3. Adjacency Matrix Example

4.3.2. *Improving Dynamic SWS Pre-Selection Using SPARQL Rewriting*

As a prerequisite for SWS composition, finding the right services to reuse and compose cross different ontologies, other-wise it is meaningless. For that, we propose a fine-grained discovery process that queries the IOs parameters of the PMs as a RDF graph using PSPARQL endpoint and none on the IOs parameters of its atomic process children that increase the search time and also the complexity. The intra-PM dependency includes the relationship between PMs according to their functionalities. Two services PM_i and PM_j are semantically interlinked if there is a semantic matching degree (Exact or Plug In matching) between the output parameters subset of PM_i and the input parameters subset of another one PM_j and defined as:

Definition 4. [Intra-PM dependency]
Let PM_1, PM_2 be two description services, and let O_{PM1}, I_{PM2} be set of outputs of PM_1 and set of inputs of PM_2. There is an intra functional dependency between PM_1 and PM_2, if ∃ an output $O_i \in O_{PM1}$ that matches an input $I_j \in I_{PM2}$ with a generalized match degree if both concepts $O_i \triangleright I_j$

Therefore, managing dynamic service dependencies should take schema-based alignment (resulting from LED) into account. In order to support the intra-PM dependency, we need to rewrite the main SPARQL query R_Q in a new one $R_{Q'}$ by using path expressions (defined in PSPARQL language) that combine generalized multiple matching patterns. The following query expresses the direct and indirect[7] intra-PM dependency with the regular expression that searches all pairs of services connected by variable path length with a sequence of an output of service s_j that feature semantically or not to the input[8] of another service s_i with regards to the generalized alignment[9] relationships, depicted in figure4 (see (a),(b) and (c)) :

[7] (at least 1), using the repeat operator plus
[8] Where "-" is the inverse operator. For example, given the RDF triple (s,p,o), we can deduce (o,p,s).
[9] equivalent or sub-concept using the repeat operator star

(a) The following query expresses the direct intra-PM dependency:

?s1 pr:hasOutput. - pr:hasInput ?s2;

(b) The following query is an extension of the previous that supports the indirect intra-PM dependency, using the repeat operator plus:

?s1 + (pr:hasOutput. - pr:hasInput) ?s2;

(c) The following query is an extension of the previous one that supports the generalized alignment relationships:

```
?s1    + ( pr:hasOutput.
         *( owl:equivalentclass | rdfs:subClassOf).
          - pr:hasInput)      ?s2;
```

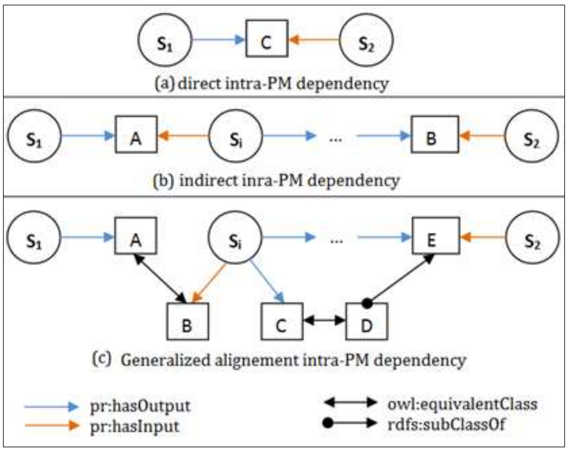

Fig. 4. Different inter-PM dependencies

On this basis, the new query R_Q' supports the IOs schema-based alignment as a disjunction of equivalence / subsomption relationships regardless the vocabulary of the ontology used where link = (owl:equivalentclass | rdfs:subClassOf) and expressed as follow:

```
Select distinct ?sj ?si Where {
?si   (pr:hasOutput.*link) ?O ;
?si   +(pr:hasInput.*link.-pr:hasOutput.*link) ?sj ;
?sj   (pr:hasInput.*link)  ?I ;
filter (((?O = O1) || (?O = O2) ...)
          && (( ?I = I1 ) || ( ?I = I2 ) ...) ) && ( ?si != ?sj)) }
```

As a pre-selection result, an ordered set of pair-wise PM denoting the initial, final and the intermediate services with no repetition by the condition (?si != ?sj) that will be generated as a M-layer graph. The complexity of determining all the composition solutions using path expression belongs to the NP space.

4.3.3. Constructing the PM Composition Graph

To find the optimal APs composition plan, we need to provide only one set of simple web services (atomic processes) with high combining ability by avoiding unused and unmanageable solutions. A set of PM was obtained that can be grouped as a set of PM cluster according to their common semantic functionalities and further transformed to M-layer graph L where a PM cluster constitute an individual graph layer and one PM could be linked semantically to one or several other PM and so, can be executed in parallel or sequencing. The layer 0 consists of a set of initial PM clusters and the final layer consists of a set of final PM clusters. For each layer (i > 0), L_i consists of a set of PM clusters that depends directly from L_{i-1} and constructed as a union of the $L_{i\ 1}$ set. The construction of the M-layer ends either when the final PM clusters are reached. The general expression for any L_i can be defined as follows:

$$L_i = \{PM_i : PM_i \notin L_j (j < i) \wedge O_{PM_i} \subseteq I_{PM_i}\}$$

In the figure5, a PM Service is rectangle and grouped with other PM Services in cluster that are represented as rectangles with rounded corners. It shows that the EventReservationService cluster is inter-connected with three services as they are grouped in two clusters, (1) the first one contains HotelReservationService and AccomodationReservationService, both of them share a common semantic inter-PM dependency parameters such as s:startDate/ s:endDate/ s:City/ s:Country that are equivalent to dbpedia:startDate/ dbpedia:endDate/ dbpedia:City/ dbpe-dia:Country (see section 4.2) and also the same output parameter (s:lodgingReservation), and (2) the second contains only the WeatherService.

4.3.4. Local Optimization for Atomic Processes Selection

The generated M-layer graph will be used for an efficient selection method targeting the identification of its relevant atomic process children according to the requirements specified in the consumer query. The commonly utilized approach is to optimize locally the PM clusters candidates independently on the other ones.

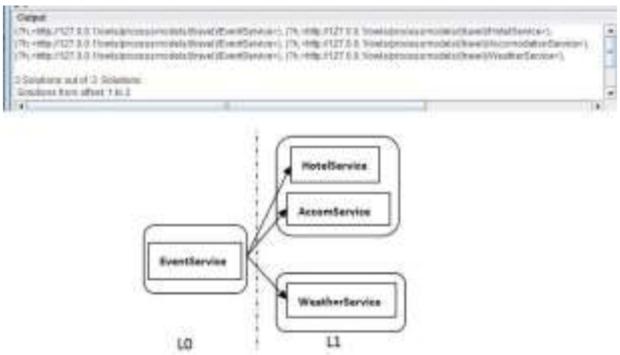

Fig. 5. M-layer Graph of the Example

So, for each cluster, only one service can be considered according to its selected atomic processes subset with higher degree of matching is chosen. The behavior of atomic processes selection is summarized by the pseudo-code listed that returns a nonempty result due to the efficient pre-selection phase. It takes as parameters: the generated M-layer graph (L), the set I_{req} of the outputs to be generated (initially the consumer query outputs, O_Q (line 2)), the set I_{av} of the available inputs (initially the consumer query inputs, I_Q (line 1)), the set Comp list of the atomic processes selected so far initially empty (line

3) and the set Sol list of the extracted sub-matrices selected so far initially empty (line 4). After initialization, the AtomSelection algorithm explores the M-graph layer, by performing a visit on each cluster (line 7) of each layer (line 5) and extracts the relevant subset of atomic processes (Service.GetAP()) of each service contained in the currently explored cluster satisfying either all inputs of atomic processes that matches I_{av} (line 12) by avoiding the unused atomic processes. Then, it computes the degree of matching(Sim_{sem}) of the relevant atomic processes for each service (line 15) and returned the most promising (higher matching value) sub-set atomic processes APs according to formula2(line 17). AtomSelection adds the subset of atomic processes to composition list (line 22), and therefore extract the sub-matrix according to its sub-set that represent a sub-workflow (line 20). Then, it updates the available inputs I_{av} by adding the outputs of the atomic processes subset (line 18), and updates the required inputs by adding the inputs of the atomic processes subset and removing all the concepts that are now available in the available inputs (lines 19). Then, AtomSelection continues on the next layers. When there are no outputs to be generated, AtomSelection selection returns the set of sub-matrices selected (line 23), which satisfies the functional consumer query. Where a sub-matrix SPM of a given graph APM = (AP, E) is a sub-graph SPM = (AP',E') where AP' $\subseteq$ AP and E' $\subseteq$ E. Let's have a process model that contains M atomic processes but only H atomic processes (H<M) matches the available inputs. Semantic similarity Sim_{AP} verifies the compatibility between the inputs of the consumer and inputs of the corresponding atomic process contained in the Process model and avoid the unmatched atomic processes. Sim_{AP} is a score computed of a set of pairs (AP_i^k .input,I_{av}) between the input parameters of i^{th} atomic process contained in cluster k from layer j and the available inputs. It measure, to what degree the inputs in the atomic process are used in the available input. We compute Sim_{AP} inspired from [24] with the following formula:

$$Sim_{AP_i}(AP_i, I_{av}) = \frac{Sim\left(AP_i^k.input, I_{av}\right)}{|AP_i.input|} \tag{1}$$

Sim_{sem} represent the global degree matching of the selected atomic processes based (1). It corresponds to the sum of Sim_{APi} of each atomic process selected according to its number that is greater than 0 and less than 1.

$$Sim_{sem}\left(AP,\ I_{av}\right) = \frac{\sum_{s=1}^{h} Sim_{APs}(AP_s, I_{av})}{h} \tag{2}$$

Let us continue with the same example, AtomSelection takes as parameters the graph L, the query inputs (i.e., In_{av}=tio:event, foaf:person, cc:creditcard, acc:hotel) and the query outputs (i.e., In_{req}=f Eventreservation, lodgingreservation, weatherconditiong), while composition is an empty set. Initially, the first layer contains one service. So, the AtomSelection algorithm extracts three APs (E1, E2 and E3) of the EventReservationService either all its inputs are available and then updates the two following sets with the outputs of the selected atomic processes I_{av}={tio:event, foaf:person, cc:creditcard, acc:hotel, dbpedia:startDate, dbpedia:endDate, dbpedia:city, dbpedia:country, s:eventReservation}, In_{req}={s:lodgingreservation, wea:weathercondition}. Then, for the second iteration, there are two similar services in the first cluster, the need to compute the semantic degree between them is required in order to select the better solution. For the HotelReservationService, it extracts two APs (H2 and H3), then it compute the degree matching that is equivalent to 1 due to the fact that all the inputs of the atomic processes are satisfying by the available equivalent inputs. For the second service, it extracts two APs (A1 and A2) with less degree of matching due to the fac that the star-ranking input of A1 cannot be fulfilled. So, it adds the first service and for the next cluster it adds W1. As a result, it returns the AP composition list with their sub-matrices; Comp={E1,E2,E3,H2,H3,W1}.

```
AtomSelection(L,I_Q,O_Q)
1:  I_av := I_Q
2:  I_req := O_Q
3:  Comp := ∅
4:  Sol := ∅
5:  for i = 1 to GetNrLayer(L) do
6:      begin
7:      for j = 1 to GetNrCluster(L, i) do
8:          begin
9:          Service=GetService(i,j)
10:         for k = 1 to Service.size() do
11:             begin
12:             atom[k]=Service.GetAP(I_av, I_req)
13:             if Service.size() = 1 then
14:                 goto marker
15:             Sim[k] = Sim_sem(atom[k], I_av)
16:             end
17:         psol= ComputeMax(Sim)
18:         marker:
19:         I_av = I_av ∪ {atom[psol].Output}
20:         I_req = {a|a ∈ {I_req ∪ atom[psol].Input} ∧ ∄b ∈ I_Av : b ▷ a}
21:         Comp = Comp ∪ {atom[psol]}
22:         end
23:     Sol = Sol ∪ {ExtractSubMtx(APM[atom[psol]])}
24:     return Sol
25:     end
```

Local Optimization based Selection

4.3.5. *Transitive Closure Composition*

We present an efficient and effective algorithm that generates dynamically the logical order of the set of atomic processes PM providing from different services that will be executed at run-time. As a result of previous section, a set of sub-matrices needed to be merged in any order to form a new square matrix, the composition dependency matrix (CMD); CMD = $\bigcup_{i=1}^{m}$ SPM_i. For instance, let's have two sub-matrices SPM_1 and SPM_2 where the vertices of each sub-matrix are defined as below V (SPM_1)={AP_{11}, AP_{12}, …, AP_{1k}} and V

$(SPM_2)=\{AP_{21}, AP_{22}, \ldots, AP_{2n}\}$. The new matrix CMD generated merges SPM_1 and SPM_2 where: V $(CMD)=\{AP_{11}, AP_{12}, \ldots AP_{1k}, AP_{21}, \ldots, AP_{2n}\}$. Then, the CMD is updated by adding the founded intra-PM dependencies between the APs. The full automatic construction of the new composite service results from the transitive closure of CMD, $CDM^{+} = CDM^{1} + CDM^{2} + \ldots + CDM^{N}$, where CMD^{N} is the matrix obtained by multiplying CDM with itself (N) times.

$$CMD = \begin{pmatrix} SPM_1 & 0 \\ 0 & SPM_2 \end{pmatrix}$$

CDM^{+} is a matrix with the same set of vertices and an edge between two vertices with path of maximum length of (N-1). Since a path of length between two vertices AP_i and AP_j exists for every vertex AP_k such that (AP_i, AP_k) and (AP_k, AP_j) are edges in E. Using the transitive closure matrix returns a deterministic result regardless the order of the merged sub-matrices. To compute faster the matrix multiplication, we use Strassen's algorithm [25] that runs in an asymptotic runtime of $\theta\ (n^{\log 7})$ or $\approx \theta\ (n^{2:81})$, making it asymptotically faster than traditional matrix multiplication, with run-time $\theta(n^3)$. The result of matrix transitive closure contains a set of paths between nodes that define the required level of connectivity of each atomic service in the new composite process with nonzero entries on its diagonal by ranking for larger values of each column in the CDM. It generates M-layer graph where the smallest number denotes to the departures nodes, and gradually locates each AP in the work-flow until the final nodes such as the APs present in the same level are executed concurrently but APs in level k should be executed before APs in level m, where m>k. As a result, a hierarchical composition graph is generated automatically.

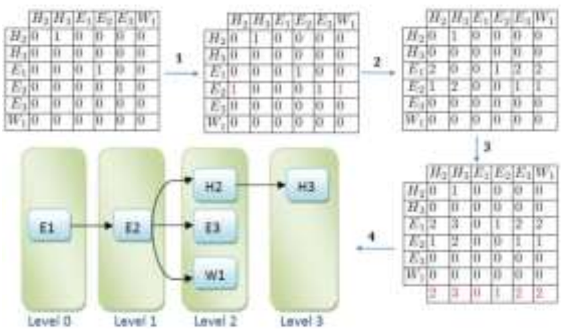

Fig. 6. Composition dependency Matrix for the travel scenario

Figure 6 shows the dependency graph matrix for the travel scenario; once the final matrix built by merging the three sub-matrices and updating the founded interdependence between E2 and H2, E2 and W1 such as E2 provides outputs that satisfy the inputs of H2 and W1 (added in red, arrow number 1). Then, the transitive closure is applied over several iterations (arrow 2 and 3). The building of the workflow (arrow 4) refers to the levels generated by the resulting matrix, starting from the lowest to the highest value. The levels 0 (E1) and 1 (E2) contain only one atomic process respectively, then the level 2 contains three concurrent atomic processes which are the direct successors of E2, thus corresponding to a complex structure, a split process between these atomic processes (H2, E3, W1). Finally, H3 is only a successor to H2 in the level 3. When a service is lost or fail in the new generated composite, another workflow can be generated if necessary by substituting with another service PM contained in the same PM cluster. It is necessary to update the composition dependency matrix.

V. Evaluation and Discussions

In order to evaluate the performance and the accuracy of the SQUIREL composition results, we created a collection (due to the unavailability of a benchmark) of OWL-S SWSs (process models) by combining and reusing existing semantic web services developed in (1) OWLS-TC 4.0 benchmark that contains 164 travel services describing a simpler composite process and none a complex one but their semantic functionalities refer to different tourism ontologies (2) Brogi'[10] collection which lists only 15 OWL-S SWSs (3) The geolocation Jena Geography Dataset (JGD) describes 203 services available at [11]. In our experiments, a set of 110 semantic Web services are managed and annotated according to the OWL-S [8] specification where some of them contain a simple composite process and others share the same functionality parameters but expressed using different ontologies. For instance, we can create a new one that allow the consumer to choose between a set of atomic processes that produces the same outputs but provided different inputs. So, we group them by using a choice control construct. Our algorithm was implemented using JavaTM JDK 1.7 and experiments were run under windows 7 PC with 64-bit operating system on a PC with an Intel Core 2 Duo E6550 at 2.33GHz and 4.00 GB of RAM. The XAMPP package was installed to create the localhost for deploying the collection of OWL-S SWSs and schema-based alignment expressed using Turtle-based syntax. In our implementation, we use CPSPARQL engine[12] that pre-select the OWL-S SWSs. Jena and ARQ to rewrite the query and Strassen algorithms to reduce the complexity of the transitive closure. The testing was conducted on the same machine via the CPSPARQL endpoint. The quality of the solutions is based on the best selected atomic processes children for each process model contained in a specific cluster from each layer. Our algorithm runs over several experimental sets of semantic web services such as the number of services vary from n =4 to 110 containing a set of atomic processes with different input and output parameters. For each run, semantic queries are generated from a randomly selected

[10] http://www.di.unipi.it/-brogi/projects/owls2pnml/owls-repository/index.html

[11] http://iserve.kmi.open.ac.uk/datasets/

[12] http://exmo.inrialpes.fr/software/psparql/

I/O parameter enabling SQUIREL to generate a distinct set of services for every simulation, we run our engine several times using the same set of services and calculated the average time. The figure 7 (the left one) shows the needed computation time for finding the optimal solution in nano-seconds. It shows a great performance as in all cases the best solution was found depending on the atomic processes children selected according to the number of inputs/outputs described in the query.

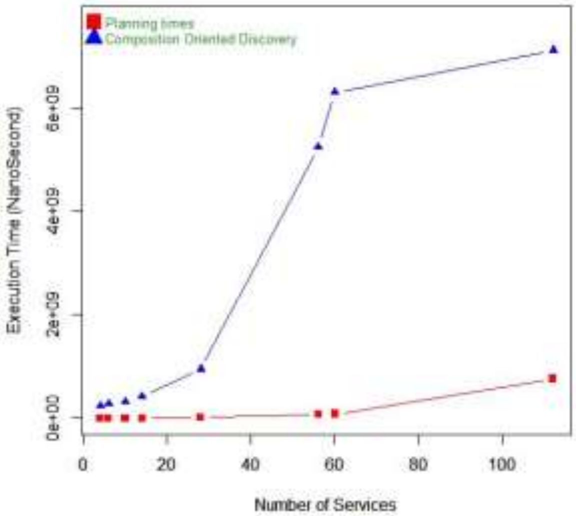

Fig. 7. Discovery-based composition

Moreover, the discovery time search (in blue) is much longer than the composition process. It enables us to explain that the computation time is not only based on the number of services but also based on the number of intra-PM dependencies when necessary. As the number of semantic web services increase, the composition solution time also grows due to the large number of equivalent combinations of services (PM clusters) that can be generated in each step. The Qsome filter finds all possible solutions with acyclic dependencies but only one solution is taken into account using an optimal algorithm. So, deleting nodes from the M-layer graph is not a simple operation. Moreover, the run-time performance does not include the construction time of the new composite service. This task is left as a future work. The resulting composite service correctness is verified manually in terms of the correctness of the execution order. Additionally, the composition process (in red) shows a great performance by using Strassen algorithm multiplication that reduces the average time response. However, discovery time search can still be reduced by decomposing the relaxed query and suggest parallelism computation for successive matrix multiplication. The figure 8 (the right one) shows the number of the inter-intra atomic processes dependency versus the number of services that was necessary to generate the optimal composition solution and performed using local optimization technique that meets fine-grained the initial request with no human intervention. It shows that the maximum length of a single workflow could contain 40 atomic processes.

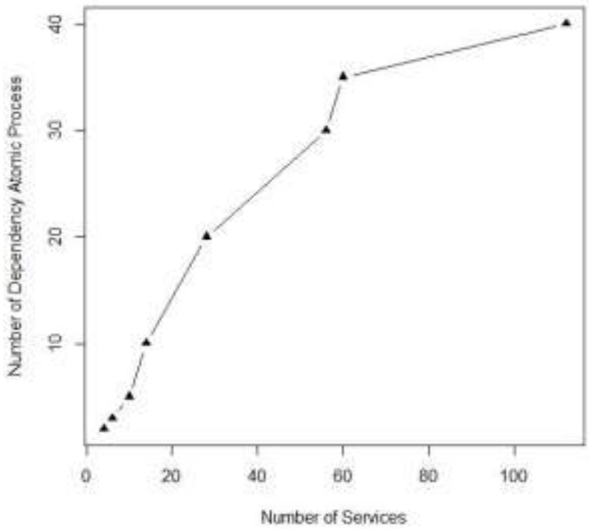

Fig. 8. Number of the inter-intra atomic processes

VI. Related Work

There are several works related [26,27,28,29,30] to the automated OWL-S Service Composition using AI-planning that translates the OWL-S process models to planning domain and the query service to a planning problem where the initial state that corresponds to the required inputs and the achieved goals that reflect the desired outputs. The both entries are submitted to a specific planner who creates a plan solution that contains a set of ordered actions that is converted to a composite service executed by OWL-S API. Other approaches [31, 32, 33, 34] treat the problem of composition as a dependency graph (tree) of services and applied a meta-heuristic search algorithm in order to extract the optimal composition scenario but none of them improve the performance in front of large amounts of services due to the redundant services. For that, [35,36] adds a set of dynamic optimization techniques over the A search algorithm. However, they treat the composition problem on simple composite processes and none on a complex process. However, very few approaches treat the problem on complex process model. We list [37] that employ an offline pre-computing semantic matching cross different ontologies to determine the dependencies within/among atomic services as well as the relationships among concept ontologies for constructing a hypergraph. A search recursive algorithm is used to analyze this hypergraph in order to discover the sets of services that are candidate to be composed given a client request. Nevertheless, this approach was tested on a small repository (ten services) and start from a predefined workflow. Another approach [38, 39] develops an algorithm that matches an I/O user query to each leaf node (atomic service) by traversing the whole processmodel through its root. If a match is found, it is added to a temporary List since it does not exist on it. If it required other inputs, it checks the next service to find other atomic processes. Finally, it returns the ordered list of atomic processes that produce the output user expected. However, it takes much time to parses the whole process models. Our composition method does not start from a predefined workflow but from the consumer query and provide a quick access to SWSs that increase the probability of finding potential SWS using SPARQL Query language.

VII. Conclusion

In this paper, we have proposed an original method for finding the optimal atomic process composition solution using SPARQL-inspired relaxation approach to improve the dynamic SWSs pre-selection without considering the entire search space. After that, it uses namely an M-layer graph so as to store the set of services as cluster of services plus the schema-based alignment metrics resulting from Linked e-Tourism Data. Therefore, in order to incrementally find the better solution, it employs a local optimization on this generated graph by considering the semantic quality of the atomic processes of each service in a cluster that's avoiding the local optimum stagnancy problem. Additionally, the proposed approach known as SQUIREL Composition Engine obtains promising results for creating dynamic packaging product but as future work, we intend to test it on more larger and complex set of process models (SWSs) by Parallelizing the computations matrix to better speed up the process of automatic composition.

References

[1] Reyhan A. Ayazlar, "Dynamic packaging applications in travel agencies," Procedia - Social and Behavioral Sciences, vol. 131, no. 0, pp. 326 – 331, 2014, 3rd World Conference on Educational Technology Re searches 2013, WCETR 2013, 7-9 November 2013, Antalya, Turkey.

[2] Jorge Cardoso and Carola Schauer, "A framework for assessing strategies and technologies for dynamic packaging applications in e-tourism," Journal of Information Technology and Tourism, vol. 9, no. 1, 2006.

[3] Jorge Cardoso, "Combining the semantic web with dynamic packaging systems," in Proceedings of the 5th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases, Stevens Point, Wisconsin, USA, 2006, AIKED'06, pp. 133–138, World Scientific and Engineering Academy and Society (WSEAS).

[4] Sonia Bilbao, Adelaida Lejarazu, and Jess Herrero, "Dynamic packaging semantic platform for tourism intermediaries," in Information and Communication Technologies in Tourism 2010, Ulrike Gretzel, Rob Law, and Matthias Fuchs, Eds., pp. 617–628. Springer Vienna, 2010.

[5] Marta Sabou, Irem Arsal, and Adrian M. P. Brasoveanu, "Tourmislod: A tourism linked data set.," Semantic Web, vol. 4, no. 3, pp. 271–276, 2013.

[6] Clara Bacciu, Angelica Lo Duca, andrea Marchetti, and Maurizio Tesconi, "Accommodations in tuscany as linked data," in Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14), Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, Eds., Reykjavik, Iceland, may 2014, European Language Resources Association (ELRA).

[7] Stefano Cresci, Andrea D'Errico, Davide Gazze, Angelica Lo Duca, Andrea Marchetti, and Maurizio Tesconi, "Towards a dbpedia of tourism: the case of tourpedia," in Proceedings of the ISWC 2014 Posters & Demonstrations Track a track within the 13th International Semantic Web Conference, ISWC 2014, Riva del Garda, Italy, October 21, 2014., Matthew Horridge, Marco Rospocher, and Jacco van Ossenbruggen, Eds. 2014, vol. 1272 of CEUR Workshop Proceedings, pp. 129– 132, CEUR-WS.org.

[8] Mark Burstein, Jerry Hobbs, Ora Lassila, Drew Mc-dermott, Sheila Mcilraith, Srini Narayanan, Massimo Paolucci, Bijan Parsia, Terry Payne, Evren Sirin, Naveen Srinivasan, and Katia Sycara, "Owl-s: Semantic markup for web services," Website, November 2004.

[9] Samur Araujo, Jan Hidders, Daniel Schwabe, and Arjen P. de Vries, "SERIMI - resource description similarity, RDF instance matching and interlinking," CoRR, vol. abs/1107.1104, 2011.

[10] Tim Berners-Lee, James Hendler, and Ora Lassila, "The semantic web," Scientific American, vol. 284, no. 5, pp. 34–43, May 2001.

[11] Patrick Hayes, "RDF Semantics," W3C Recommendation, 2004.

[12] Deborah L. McGuinness and Frank van Harmelen, "Owl - web ontology language," 2004, http://www.w3.org/2004/OWL/.

[13] Eric Prud'hommeaux and Andy Seaborne, "Sparql query language for rdf," Latest version available as http://www.w3.org/TR/rdf-sparql-query/, January 2008.

[14] Marcelo Arenas and Jorge Perez, "Querying semantic web data with sparql," in Proceedings of the Thirtieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, New York, NY, USA, 2011, PODS '11, pp. 305–316, ACM.

[15] Faisal Alkhateeb, Jean-Fran çois Baget, and Jerome Euzenat, "Extending SPARQL with regular expression patterns (for querying RDF)," J. Web Sem., vol. 7, no. 2, pp. 57–73, 2009.

[16] Ruben Verborgh, Miel Vander Sande, Pieter Colpaert, Sam Coppens, Erik Mannens, and Rik Van de Walle, "Web-scale querying through Linked Data Fragments," in Proceedings of the 7th Workshop on Linked Data on the Web, Apr. 2014.

[17] Jos Mara Garca, David Ruiz, and Antonio Ruiz-Corts, "Improving semantic web services discovery using sparql-based repository filtering," Web Semantics: Science, Services and Agents on the World Wide Web, vol. 17, no. 0, 2012.

[18] Christian Bizer, Tom Heath, and Tim Berners-Lee, "Linked data - the story so far," International Journal on Semantic Web and Information Systems (IJSWIS), 2009.

[19] Tom Heath and Christian Bizer, Linked Data: Evolving the Web into a Global Data Space, 2011.

[20] Anja Jentzsch, Robert Isele, and Chris Bizer, "Silk-generating rdf links while publishing or consuming linked data," in 9th International Semantic Web Conference (ISWC2010), November 2010.

[21] Ghislain Atemezing and Raphael¨ Troncy, "Comparing vocabularies for representing geographical features and their geometry," in ISWC 2012, 11th International Semantic Web Conference, Terra Cognita Workshop, Volume 901, November 11-15, 2012, Boston, USA,

Boston, ETATS-UNIS, 11 2012.
[22] Reinhard Diestel, Graph Theory, Number 173 in Graduate Texts in Mathematics. Springer, 1997.
[23] Saida Boukhedouma, Mourad Oussalah, Zaia Alimazighi, and Dalila Tamzalit, "Service based cooperation patterns to support flexible inter-organizational workflows," International Journal of Information Technology and Computer Science(IJITCS), vol. 6, no. 4, pp. 1–18, Mar. 2014.
[24] Barry Norton and Steffen Stadtmller, "Scalable discovery of linked services," in Proceedings of the Fourth International Workshop on REsource Discovery (RED 2011) co-located with the 8th Extended Semantic Web Conference, ESWC2011, May 30, 2011., 2011, pp. 6– 21.
[25] Mithuna Thottethodi, Siddhartha Chatterjee, and Alvin R. Lebeck, "Tuning strassen's matrix multiplication for memory efficiency," in In Proceedings of SC98 (CD-ROM, 1998.
[26] Evren Sirin, Bijan Parsia, Dan Wu, James Hendler, and Dana Nau, "Htn planning for web service composition using shop2," Web Semant., vol. 1, no. 4, pp. 377–396, Oct. 2004.
[27] Sheila A. McIlraith and TC Son, "Adapting golog for composition of semantic web services," in Proceedings of the Eighth International Conference on Knowledge Representation and Reasoning (KR2002), April 22-25 2002, pp. 482–493.
[28] Ourania Hatzi, Dimitris Vrakas, Nick Bassiliades, Dimosthenis Anagnostopoulos, and Ioannis P. Vlahavas, "The PORSCE II framework: using AI planning for automated semantic web service composition," Knowledge Eng. Review, vol. 28, no. 2, pp. 137–156, 2013.
[29] Matthias Klusch and Andreas Gerber, "Evaluation of service composition planning with owls-xplan," in Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, Washington, DC, USA, 2006, WI-IATW '06, pp. 117–120, IEEE Computer Society.
[30] V. Portchelvi and V. Prasanna Venkatesan, "A goal-directed orchestration approach for agile service composition," International Journal of Information Technology and Computer Science(IJITCS), vol. 7, no. 3, pp. 60–67, Feb. 2015.
[31] Eduardo Goncalves da Silva, Lus Ferreira Pires, and Marten van Sinderen, "Towards runtime discovery, selection and composition of semantic services.," Computer Communications, vol. 34, no. 2, pp. 159–168, 2011.
[32] S. Kona, A. Bansal, M.B. Blake, and G. Gupta, "Generalized semantics-based service composition," in Web Services, 2008. ICWS '08. IEEE International Conference on, Sept 2008, pp. 219–227.
[33] Yixin Yan, Bin Xu, and Zhifeng Gu, "Automatic service composition using and/or graph," in E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing, E-Commerce and E-Services, 2008 10th IEEE Conference on, July 2008, pp. 335–338.
[34] F. Lecué, E. M. Goncalves da Silva, and L. Ferreira Pires, "A framework for dynamic web services composition," in 2nd ECOWS Workshop on Emerging Web Services Technology (WEWST07), Halle, Germany, Germany, November 2007, CEUR Workshop Proceedings.
[35] Pablo Rodriguez-Mier, Manuel Mucientes, and Manuel Lama, "Automatic web service composition with a heuristic-based search algorithm," in IEEE International Conference on Web Services, ICWS 2011, Washington, DC, USA, July 4-9, 2011. 2011, pp. 81–88, IEEE Computer Society.
[36] Pablo Rodriguez-Mier, Carlos Pedrinaci, Manuel Lama, and Manuel Mucientes, "An integrated semantic web service discovery and composition framework," IEEE Transactions on Services Computing, p. aceptado, 2015.
[37] Antonio Brogi, Sara Corfini, and Razvan Popescu, "Semantics-based composition-oriented discovery of web services," ACM Trans. Internet Techn., vol. 8, no. 4, 2008.
[38] D. Paulraj, S. Swamynathan, and M. Madhaiyan, "Process model-based atomic service discovery and composition of composite semantic web services using web ontology language for services (OWL-S)," Enterprise IS, vol. 6, no. 4, pp. 445–471, 2012.
[39] D. Paulraj, S. Swamynathan, and M. Madhaiyan, "Process model ontology-based matchmaking of semantic web services," Int. J. Cooperative Inf. Syst., vol. 20, no. 4, pp. 357–370, 2011.

Authors' Profiles

Afaf Merazi: received the Engineer Degree and the Magister degree in Computer Science from DJILLALI LIABES University of Sidi Bel Abbes (Algeria) in 2005 and 2007, respectively. Actually, she is a PhD student in Computer Science Department at the DJILLALI LIABES University of Sidi Bel Abbes (Algeria). Her academic research interests include Semantic Web (Services), Tourism Industry, Discovery-based Composition, Linked Data, Semantic Querying, Ontology Alignment.

Mimoun Malki: is graduated with Engineer Degree in Computer Science from National Institute of Computer Science, Algiers (Algeria) in 1983. He received the Magister degree and the PhD degree in Computer Science from DJILLALI LIABES University of Sidi Bel Abbes (Algeria) in 1992 and 2002, respectively. He was a Senior Lecturer in Computer Science Depart-ment at the same university from 1986 to 2002. Currently, he is a Professor at the High School of Computer Science of Sidi Bel Abbes (ESI- Algeria). His research interests include databases, ontology engineering, web based information systems, semantic web (services), Linked Open Data and web re-engineering.

How to cite this paper: Afaf Merazi, Mimoun Malki,"SQUIREL: Semantic Querying Interlinked OWL-S traveling Process Models", International Journal of Information Technology and Computer Science(IJITCS), vol.7, no.12, pp.30-39, 2015. DOI: 10.5815/ijitcs.2015.12.04