

NEW ALGORITHM

Input: an incoming query Q_{in} , which includes a set of abstract services ($A_1..A_n$) and a set of user integration requirements ($R_1..R_m$).

Output: the best reusable service composition in accordance with the incoming query.

```
1: function reuseQueries ( $Q_{in}$ )
2:    $L_Q \leftarrow \emptyset$ 
3:    $L_Q \leftarrow \text{searchForQueries} (Q_{in}, \text{"equivalent"})$ 
4:   if  $L_Q \neq \emptyset$  then
5:      $p \leftarrow L_Q.\text{next} ()$ 
6:      $L_{RC} \leftarrow \emptyset$ 
7:     while  $p \neq \text{null}$  do
8:        $L_{RC} \leftarrow L_{RC} \cup p.\text{getOnlineCompositions} ()$ 
9:        $p \leftarrow L_Q.\text{next} ()$ 
10:    end while
11:     $L_{RC} \leftarrow \text{validateCompositions} (L_{RC})$ 
12:    storeResults ( $Q_{in}, L_{RC}$ )
13:    return first ( $L_{RC}$ )
14:  end if
15:   $L_Q \leftarrow \text{searchForQueries} (Q_{in}, \text{"superset"})$ 
16:  if  $L_Q \neq \emptyset$  then
17:     $p \leftarrow L_Q.\text{next} ()$ 
18:     $L_{RC} \leftarrow \emptyset$ 
19:    while  $p \neq \text{null}$  do
20:       $L_{RC} \leftarrow L_{RC} \cup p.\text{getOnlineCompositions} ()$ 
21:       $L_{RC} \leftarrow \text{projectCompositions} (L_{RC})$ 
22:       $p \leftarrow L_Q.\text{next} ()$ 
23:    end while
24:     $L_{RC} \leftarrow \text{validateCompositions} (L_{RC})$ 
25:    storeResults ( $Q_{in}, L_{RC}$ )
26:    return first ( $L_{RC}$ )
27:  end if
```

```

28:   $L_Q \leftarrow \text{searchForQueries} (Q_{in}, \text{"subset"})$ 
29:  if  $L_Q \neq \emptyset$  then
30:       $p \leftarrow L_Q.\text{next} ()$ 
31:       $L_{RC} \leftarrow \emptyset$ 
32:      while  $p \neq \text{null}$  do
33:           $L_{RC}' \leftarrow \emptyset$ 
34:           $L_{CQ} \leftarrow \emptyset$ 
35:           $L_{RC}' \leftarrow L_{RC}' \cup p.\text{getOnlineCompositions} ()$ 
36:           $L_{CQ} \leftarrow \text{searchForComplementaryQuery} (p)$ 
37:           $p' \leftarrow L_{CQ}.\text{next} ()$ 
38:          while  $p' \neq \emptyset$  do
39:               $L_{RC}'' \leftarrow \emptyset$ 
40:               $L_{RC}'' \leftarrow p'.\text{getOnlineCompositions} ()$ 
41:               $L_{RC} \leftarrow L_{RC} \cup \text{combineCompositions} (L_{RC}', L_{RC}'')$ 
42:               $p' \leftarrow L_{CQ}.\text{next} ()$ 
43:          end while
44:           $p \leftarrow L_Q.\text{next} ()$ 
45:      end while
46:       $L_{RC} \leftarrow \text{validateCompositions} (L_{RC})$ 
47:       $\text{storeResults} (Q_{in}, L_{RC})$ 
48:      return  $\text{first} (L_{RC})$ 
49:  end if
50: end function

```

The idea behind this algorithm is given an incoming query Q_{in} , (i) to find a set of previous stored queries that could be reused in order to answer the user request, (ii) to identify all available compositions associated to these previous queries, and (iii) to select the best composition for answering the query.

First, the algorithm searches for previous queries that could be used to answer the incoming query (line 2). The method $\text{searchForQueries} (Q_{in})$ executes a stored procedure that retrieves the reusable queries according to their type. For example, *equivalent* queries have priority over *superset* queries, and *superset* queries have priority over *subset* queries. Thus, the set of queries retrieved belongs to single type respecting this priority. *Second*, the algorithm iterates (lines 5

to 22) over the set of queries (L_Q) performing the necessary operations to reuse them. If L_Q includes *equivalent* queries or *superset* queries, (i) the algorithm filters the available compositions associated to each query in this set, and (ii) it projects the part of the composition that can answer the user request. Otherwise, the algorithm searches for queries that can be complementary (*searchForComplementaryQuery*, line 12) and iterates on them (lines 14 to 19) combining the available compositions (lines 16 and 17). *Finally*, the reusable compositions are validated (*validateCompositions*, line 23), the results stored for a next integration request (*storeResults*, line 24), and the best composition, which satisfies the user requirements, is returned (line 25).