Report 26 April

What are the *derived SLA* and *integration SLA*? How do you structure them to reuse and optimize a further integration? Where can we find them?

The *derived SLA* is a type of SLA that includes information about the query, user preferences, user cloud subscriptions, the different data services and data processing services that can be used to answer the query, the compositions generated using data services, the compositions that were executed while meeting the user preferences and subscriptions, and where the results are delivered. The *derived SLA* is build dynamically during the whole integration process. When the results are delivered to the user, fulfilling his/her preferences and in accordance with his/her cloud subscriptions, an *integration SLA* is created based on the *derived SLA*. It will include the query, user preferences, user cloud subscriptions, the data services and data processing services that were used to answer the query, the compositions that were executed while meeting the user preferences and subscriptions, and where the results are delivered. The *integration SLA* is a sub-part of a *derived SLA*. *Integration SLA* and *derived SLA* are stored in a SLA registry in order to be reused in a further integration.

How does the whole process work?

Given a query, user preferences and specific user cloud subscriptions, the integration steps are:
Building the *derived SLA* including the query, user preferences and user cloud subscriptions.
Looking for previous *integration SLA*. We will search on our SLA registry for previous *integration SLA* that can be matched with the actual user query, preferences and cloud subscriptions. If a complete match is found, the information about used data services, generated and executed compositions are reused. If a partial match is found, for example, we have found an *integration SLA* with the same query, but with different preferences and cloud subscription restrictions, the data services that were listed as possible match with the query are reused.
Looking for data services. If no match can be found in the previous *integration SLA*s, we will proceed selecting data services that can be matched with the query. Here, we should use and extend the *Rhone*'s service selection step. Actually, the *Rhone* select services based on their definition, a structural match of service name and types of variables. I believe we should extend this process allowing a semantic match of services and variables. I think it should be done using ontologies. The result of this step is a set of services that can answer the query concepts in totality or partially.
Filtering data services. Data services selected in the previous step are filtered based on the user preferences and user cloud subscriptions. Here we also need to extend the *Rhone* filtering process. *Rhone* does not consider the restrictions imposed by the user cloud subscriptions, it just take into account the user preferences.

We have discussed in the last meeting that data services would be filtered considering the *SLAs* deduced from user's *SLAc*. However, reflecting today, I was not able to see how to automatically generate *SLAs* from different user's *SLAc*. The first reason is the *SLAs* should describe quality conditions the data service is delivered. Then, I believe this kind of information should be specified by who is providing the service (for example, the cloud that hosts the service, the owner of the service). Another reason is that the manipulation and combination of these *SLAs* could be not efficient considering that for each user cloud subscription a *SLAs* should be produced for each data service selected in the previous phase. This means in the worst case, if we have 10 services and 3 cloud subscriptions, we will produce 30 *SLAs*. I really do not know if it is interesting or not.

Concerning this issue, I believe that the cloud provider should define and furnish the different *SLAs* associated to each service. Thus, the filtering process is done matching the user preferences and user cloud subscriptions with the quality condition that a service is delivered specified in the *SLAs*.
Producing rewritings (service compositions). Given only the services that were filtered in the previous step, they are combined and checked if they are a rewriting of the query or not. Here we will use the another part of the *Rhone* for combining and producing rewritings. However, we need to extend it to consider the user cloud subscription while verifying and validating a rewriting.
Executing the integration. The best data processing service to execute the composition is selected considering the cost for retrieving, integrating and transferring data. The compositions are executed while the preferences and subscription restrictions are being satisfied.

Deliverables

- Examples of the different contracts (*derived, integration, SLAs and SLAc*)
- Matching example considering the SLAs
- Approach for matching the *derived SLA* with the *integration SLA*
- Extending *Rhone* selection process to include the semantic matching of services
- Algorithm for matching the user preferences and cloud subscription with the *SLAs* of the different services while selecting and rewriting the query (another extension to *Rhone*)

- Algorithm to evaluate the best location to execute the query considering the user cloud subscription and query economic cost
- Improve and detail the general process

Approach for matching SLAs

The objective of this approach is to identify and select the best *integration SLA* that matches with the actual query and user requirements in order to be reused during the integration process. The approach is divided in three matching steps: query matching, preferences matching and cloud subscriptions matching.

Query matching: given the user query, the objective is to match it with previous queries submitted to an integration. Previous queries can be retrieved from the *integration SLAs* stored in our registry. Three types of query matching matching can occur: (i) full match in which the queries includes the same concepts and semantics; (ii) a partial match in which the actual query is contained in a previous query; and a (iii) a partial match in which the actual query contains part of a previous query. The (i), (ii) and (iii) is the priority order while choosing a SLA. During the matching process if a full match is found, the partial matches are discarded. In the same sense, if no full match is found and the first case of partial match is found, the second case of partial match is discarded. Partial matches are selected considering their order. The order is the number of abstract services of the actual query that is covered by a previous query. The *integration SLAs* with the highest order are selected. The result of this step is a set of *integration SLAs* that can be reused in the integration process and that are closest as possible to the actual query.

Preferences matching: the idea in the step is to filter the *integration SLAs* selected in the previous step considering the user preferences. Three types of preferences matching can occur: (i) full match when the preferences are the same; (ii) a partial match in which the actual preferences are more restrict and specific than the preferences of previous query; and (iii) a partial match in which the previous preferences are more restrict and specific than the preferences of actual query.

Subscriptions matching: the idea is to filter the set of *integration SLAs* obtained from the previous step considering the different subscriptions the user has with the clouds. Three types of subscriptions match can occur: (i) a full match in which the user subscriptions are the same; (ii) a partial match in which the the actual user has the subscriptions to the same cloud as the user of a previous integration, but with different restrictions; and (iii) a partial match in which the actual user subscriptions and the user subscriptions of a previous integration are different, but they have subscriptions in common. The *integration SLA* selection priority is (i), (ii) and (iii), respectively. The result is one single *integration SLA* that is going to be reused in the integration.

Reusing an *integration SLA* and updating the *derived SLA*

Given an *integration SLA* selected considering the matching approach, the *derived SLA* is updated using the *integration SLA* information. Depending on the type of match occurred while selecting the *integration SLA*, different actions should be taken.

Case 1: query full match, user preferences full match and subscriptions full match. In this case, the *integration SLA* is completely reused. The used data services and data processing services, and the executed compositions are reused for this new integration request. These information is added to the *derived SLA* created to this new request.

Case 2: query full match, no match on user preferences and no match on subscriptions. In this case, the *integration SLA* is partially reused. Only the data services that were selected to this query are reused. This information is added to the *derived SLA* created to this new request. The filtering and rewriting process should be done for this new request. I am assuming that the *integration SLA* has an indication to another XML file that includes the query and the complete set of data services that can be used in the rewrite.

Case 3: query full match, partial match (1) on user preferences and no match on subscriptions. In this case, the *integration SLA* is partially reused. The partial match (1) is the one in which the actual user preferences are more restrict and specif than the preferences on the selected *integration SLA*. The data service filtering and rewriting process is done again but only over the data services and rewritings that were used and produced for the previous *integration SLA*. The new filtered services and rewritings are included in the *derived SLA*.

Case 4: query full match, partial match (2) or no match on user preferences and no match on subscriptions. In this case, the *integration SLA* is partially reused. The partial match (2) is the one in which the previous user preferences are more restrict and specif than the actual preferences. Only the data services that were selected to this query are reused. This information is added to the *derived SLA* created to this new request. The filtering and rewriting process should be done for this new request. I am assuming that the *integration SLA* has an indication to another XML file that includes the query and the complete set of data services that can be used in the rewrite.

Case 5: query full match, full match on user preferences and partial match (1 and 2) or no match on subscriptions. In this case, the *integration SLA* is partially reused. The partial match (1) on subscriptions is the one in which the actual user subscriptions and the previous identified in the *integration SLA* are in the same clouds, but with different restrictions over them. The partial match (2) occurs when the actual user subscriptions and the subscriptions on the *integration SLA* are different in terms of clouds and restrictions on them. No matter the type of partial match or no match on subscriptions, the used services and the rewritings produced are reused. A new filtering process is done over this reduced set of services and rewritings. The new filtered services and rewritings are included in the *derived SLA*.

Case 6: query full match, partial match (1) on user preferences and full match or partial match (1 and 2) or no match on subscriptions. In this case, the *integration SLA* is partially reused. No matter the type of partial match or no match on subscriptions, when a partial match (1) occurs in the preferences, the used services and the rewritings produced are reused. A new filtering process is done over this reduced set of services and rewritings. The new filtered services and rewritings are included in the *derived SLA*.

Case 7: query full match, partial match (2) on user preferences and full match or partial match (1 and 2) or no match on subscriptions. In this case, the *integration SLA* is partially reused. No matter the type of partial match or no match on subscriptions, when a partial match (2) occurs in the preferences, the complete set of selected services are reused. The filtering process is done over the services and all rewritings are produced again. The filtered services and rewritings are included in the *derived SLA*.