QoS-Enhanced Broker for Composite Web Service Selection

Salem Chakhar
CRAD, ESAD, University Laval
Pavillon Félix Antoine Savard, Québec City, Québec G1K 7P4, Canada
Email: salem.chakhar@crad.ulaval.ca

***Abstract*—The paper proposes a layered system for web service composition. The input for the system is the specification of the desired service, including both functional and non-functional requirements. The composition operation takes as input a generic composition graph defined based on the functional requirements. Then, the set of potential compositions is identified based on "hard" non-functional requirements. This set is represented in terms of an extended version of the composition graph that permits to take into account the different BPEL constructors. Finally, best composition(s) are identified by solving a bi-objective shortest path problem on the transformed composition graph.**

***Keywords*-web service; service selection; service composition; quality of service;**

I. Introduction

Web service composition attempts to take advantage of currently existing web services to provide a new service that does not exist on its own [1]. Therefore, in order to have a more complex service, we can use some semantically related simpler web services and execute them in such a way that the whole set provides the desired service [1]. One important issue within web service composition is related to the selection of the most appropriate one among the different possible compositions. Most of current works use successive evaluation of different (non functional) aspects in order to attribute a general "level of quality" to different composite web services and to select the "best" one from these services. In these works, the evaluation of composite web services is based either on a single Quality of Service (QoS) attribute (such as availability, response time, etc.) or, at best, on a weighted sum of several quantitative attributes.

More advanced web service composition proposals are based on ontology [2], multi-agent systems [3], genetic algorithms [4], stochastic optimization algorithm [5], linear programming [6], query languages [7], and multicriteria evaluation [8][9][10][11] A recent survey of web service composition approaches is available in [12].

The goal of this research is to propose a layered system for web service composition. The proposed system uses multicriteria evaluation to identify the composite service that responds better to the client functional and non-functional requirements. The input for the system is the specification of the desired service, including both functional and non-functional requirements. The composition operation takes as input a generic composition graph defined based on the functional requirements. Then, the set of potential compositions is identified based on "hard" non-functional requirements. This set is represented in terms of an extended version of the composition graph that permits to take into account the different BPEL constructors. Finally, "best" composition(s), in terms of cost and "soft" non-functional requirements, are identified by solving a bi-objective shortest path problem on the transformed composition graph.

The paper geos as follows. Section II sets the background. Section III introduced the composition approach and system architecture. Sections IV, V and VI detail the three phases of the composition approach. Section VII concludes the paper.

II. Background

The input for web service composition is a set of specifications describing the capabilities of the desired service. These specifications can be decomposed into two groups: (i) functional (FC) requirements that deal with the desired functionality of the composite service; and (ii) non-functional (NFC) requirements that relate to issues like cost, performance and availability. These specifications need to be expressed in an appropriate language. The framework presented here uses an extended version of Ontology Web Language (OWL) proposed in [13] for expressing functional requirements and the QoS for non-functional requirements.

A. Service Type and Instance

In this paper, as in [13], we differentiate between web service *types*, which are groupings of similar (in terms of functionality) web services, and the actual web service *instances* that can be invoked. We believe as [13] that the separate representation of web service type definition from instance definition helps in handling different requirements, and different means to optimize them, and allows us to work efficiently with large collection of web services. The web service types and instances can be advertised in a registry.

Following [10], a web service type is defined as follows.

Definition 1: A **web service type** S_i is a tuple $\langle F_i, Q_i, C_i \rangle$, where:

- F_i is a description of the service's functionality,
- Q_i is a specification of its QoS attributes, and
- C_i is its cost specification.$\Diamond$

Each web service type S_i has a unique functionality F_i. In turn, the same functionality may be supported by different service types. We denote by $I_i = \{A_1^i, A_2^i, \cdots, A_{n_i}^i\}$

the set of service instances associated with S_i; n_i is the number of instances for service type S_i.

The QoS attributes can be subdivided into two groups :

- *Hard attributes*: These correspond to QoS attributes for which some obligatory constraints are imposed by the client. Instances that fail to meet these constraints are automatically eliminated.
- *Soft attributes*: These correspond to QoS attributes that should be optimized, i.e. maximized or minimized, according to the user preferences.

Hard QoS attributes will be used in the second phase of the proposed composition approach to eliminate services instances that fail to meet the hard QoS requirements. Soft QoS attributes will be used to compare the different potential compositions. In the rest of the paper, we denote by Q^h the set of hard attributes and by Q^s the set of soft attributes with $Q_i = Q^s \cup Q^h$ and $Q^s \cap Q^h = \emptyset$.

B. Composite Service Type and Instance

Composite web service type and instance are defined as follows.

Definition 2: A **composite service type** is a tuple $\langle S, R, Q, C \rangle$, where:

- $S = \{S_1, \cdots, S_n\}$ is a collection of n service types,
- R is a specification of the invocation relationships among service types in S,
- Q is a specification of its QoS attributes, and
- C is its cost specification.$\diamondsuit$

The functionality of the composite service can be retrieved from set R. Naturally, the specification Q of QoS attributes and cost specification C of a composite service are defined based on the QoS attributes Q_i and cost C_i specifications of simple services implied in the composition. Accordingly, appropriate aggregation rules need to be defined and used to combine QoS and cost attributes of simple services into specifications that apply to the composite service as a whole.

Definition 3: A **composite service instance** J_k is a collection of service instances $(A^1_{(k_1)}, \cdots, A^i_{(k_i)}, \cdots, A^n_{(k_n)})$ with $A^i_{(k_i)} \in A_i$ for $i = 1$ to n and $k_i \in \{1, \cdots, n_i\}$.$\diamondsuit$

In the rest of this paper we denote by J the set of all compositions. Once created, a composite service, as any simple web service, can be transformed into a workflow and then deployed, discovered and invoked.

C. Service Composition Problem Description

Service composition problem can be stated as follows. Given a set of web service types and the set of instances for each type, along with the specifications of a new service, create an executable plan (i.e. workflow) that satisfies the following objectives:

- provides the desired functionality, (*obj1*)
- verifies the user's QoS constraints, (*obj2*)
- minimizes the composite service cost, and (*obj3*)
- maximizes the composite service QoS. (*obj4*)

The first objective *obj1* is achieved by constructing a composition graph according to the functional requirements. The second objective *obj2* is achieved by eliminating all instances that fail to meet constraints associated with QoS attributes in Q^h. Objectives *obj3* and *obj4* are related to the evaluation and selection of compositions. These two objectives are conflicting and an appropriate algorithm need to be advised to dealt with them.

III. Composition Approach and Architecture

A. Service Composition Approach

The key elements of the proposed approach are the composition graph, potential executable plans and executable plan.

1) Composition Graph: The FC requirements provided by the client can be used to construct an abstract representation of the composite web service. Let K be a composite web service defined as $\langle S, R, Q, C \rangle$. Then, the invocation relationships in R may be represented by a connected and directed graph $G = (S, R)$ where:

- $S = \{S_i, S_j, \cdots, S_n\}$ is a set of services types,
- $R = \{(S_i, S_j) : S_i, S_j \in S \wedge S_i$ can invoke $S_j\}$.

The graph G is called the *composition graph*.

2) Potential Executable Plans: The set J of the composite service instances is obtained by replacing each service type in the composition graph by its instances using a set of transformation rules. The transformation operation has two objectives: (i) include the different semantics of BPEL constructors, and (ii) eliminate service instances that fail to meet QoS constraints associated attributes in Q^h. Each element of set J represents a *potential executable plan*.

3) Executable Plan: Among the different potential executable plans in set J only one—called *executable plan* and denoted J^*—should be selected and transformed to a workflow for effective execution.

The composition operation starts by user specification of FC and NFC requirements and leads to an executable plan J^* that can be handed off to runtime environment for execution. The proposed approach to support the composition operation is composed of three phases:

1) *Logical composition*: First, the FC requirements provided by the user are used to generate the composition graph G.
2) *Physical composition*: Second, the composition graph is transformed to obtain the set J of potential executable plans.
3) *Evaluation and selection*: Third, the different potential executable plans are evaluated and compared in order to select one executable plan, namely J^*. The latter is then transformed into a workflow and then deployed, discovered and invoked.

The first phase permits to reply to the first objective (*obj1*) of the service composition problem since it permits to handle the different FC requirements. The second phase permits to handle the different constraints associated attributes in Q^h and so it responds to the second objective (*obj2*). The third phase uses a bi-objective algorithm to tradeoff objectives *obj3* and *obj4*.

B. System Architecture

The service composition approach is implemented by a layered system called QoSeBroker (for QoS-Enhanced Broker). The architecture of QoSeBroker is given in Figure 1. This system is composed of three layers; each layer supports one phase of the service composition approach.

1) Logical Composition Layer: This layer is responsible for logical composition. It transforms the FC requirements provided by the client into a composition graph G representing a new service type (*obj1*). The specifications of available service types are stored into the service registry. The domain information are defined in terms of an ontology. When a new service needs to be created, the client provides the specification of the desired service to the Logical Composer Module. The latter then explores the registry and uses agent-based system module to create a composition graph that meets the specified requirements.

2) Physical Composition Layer: This layer takes as input the composition graph G and generates the set of potential executable plans J. The basic idea consists in replacing each service type in G by its instances while applying some transformation rules. The transformation operation is handled by the Composition Graph Transformation Module. The transformation operation requires that the QoS of each implied web service is evaluated independently. This is supported by the QoS Evaluation Module. The Physical Composer layer uses the Dominance Analysis Module to eliminate instances that fail to meet the QoS constraints (*obj2*).

3) Evaluation and Selection Layer: The input for this layer is the set J represented in terms of the transformed composition graph. The Evaluator and Selector Module uses this graph to identify the one that tradeoffs at test the cost (*obj3*) and the soft QoS attributes (*obj4*). The best executable plan $J*$ can then be deployed into a runtime infrastructure using a workflow engine such as WebSphere Process Choreographer.

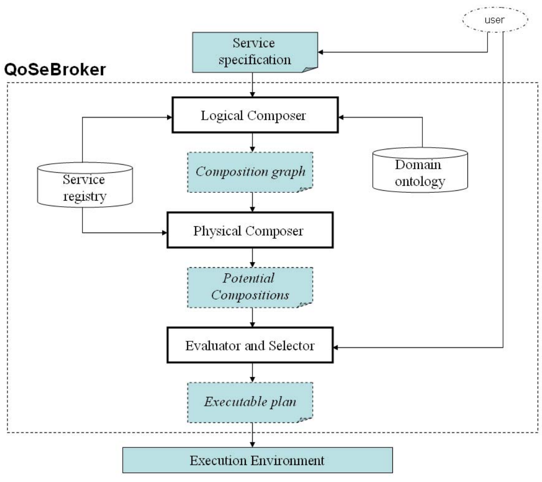

Figure 1. QoSeBroker architecture

IV. Logical Composition

The architecture of logical composition layer is given in Figure 2. The logical composition implies the following steps. First, the specification of the required service need to be provided by the user. Second, the Matchmaker module queries the service registry and identifies available services. Third, the Planner Module uses planning techniques and the agent-based Graph Constructor Module to create the composition graph.

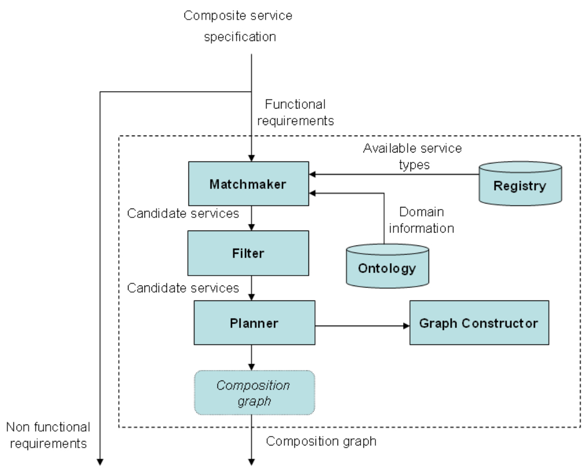

Figure 2. Architecture of logical composition layer

A. Representation of Service Types

Service types need to be described in a high-level and abstract manner. This enables their automatic discovery and composition of desired functionality. In this paper, we adopt an extended version of the OWL-S proposed in [13] to create the domain models. OWL-S specifies an upper ontology of services that defines the structure of a service description. OWL-S defines that a service presents a *ServiceProfile* (what the service does), is described by a *ServiceModel* (how it works) and supports a *ServiceGrounding* (how to access it) (see Figure 3).

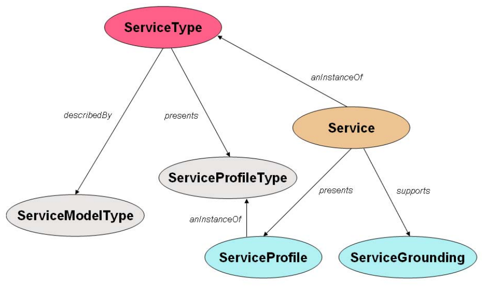

Figure 3. Extended OWL-S upper ontology [13]

The enhanced OWL-S defines a *ServiceType* class hierarchy in addition to the service hierarchy. The *ServiceProfile* of an instance points to the corresponding *ServiceProfileType* for mandatory portion common to all instances. It could, however, add its own precondition and effects. Similarly, it may support additional outputs as well as inputs, in which case it specifies the default values of additional inputs so that the compositions done using *ServiceProfileType* remain valid.

Obviously, a composition should remain valid when any of its instances is selected. This holds if the precondition of the service type is more specific than precondition of some or all of its instances and its effects is more general than effect of any of its instances. This means that the two following conditions should hold [13]: (i) the precondition of the service type entails the precondition of the service instance so that the latter is satisfied whenever the former is, and (ii) the postcondition of the service instance entails the postcondition of the service type so that the latter is satisfied whenever the former is.

Within the initial OWL-S, the FC of a web service are expressed through IOPE (Inputs-Outputs-Preconditions-Effects), which captures the transformation performed by this service. OWL-S can also be used to represent NFC through profile attributes, which may contain parameters other than the functional IOPE. In the extended OWL-S upper ontology, the FC are represented in *ServiceProfile-Type*. The *ServiceProfile* of an instance inherits these FC from the *ServiceProfileType* and adds the NFC to it.

B. Specification of Desired Composite Service

In order to create a new service, the client should describe the desired FC and NFC requirements. In this paper, as in [13], we adopt OWL-S for representing the functional requirements, in IOPE terms, of the composite service. The requirements are processed incrementally. The preconditions and effects are logical terms and expressions, and are used during planning in logical composition. The inputs and outputs are expressions involving general data types (e.g. integers, strings, algebraic expressions) which are used during instance selection and flow concretization in the physical composition phase.

C. Identification of Candidate Services

Construction of the composition graph requires first the identification of candidate services. This task is handled by the Matchmaker Module. This module matches the preconditions of a web service with the effects of another up front during filtering. Note that matchmaking in this level is based only on FC requirements. The filtering operation is supported by the Filter Module. This modules removes irrelevant web services based on the goals specified by the user [13]. Relevant services are those that can either contribute (i) to the goals (at least one effect unifies with a goal) or (ii) to the preconditions of any service which can potentially contribute to the goal.

D. Construction of Composition Graph

The Planner Module uses planning techniques and the agent-based system, Graph Constructor Module, to create the composition graph. This model uses the solution proposed in [8]. The aim of this tool is to assemble the service types identified by the Matchmaker using the different BPEL constructors. The composition operation generally starts by the invocation of a main web service and ends by another specific web service. We design these two specific web services by SWS (*Start Web Service*) and TWS (*Terminal Web Service*). For instance, Figure 4 presents a composition graph that implies six service types (S_1, S_2, S_3, S_4, S_5 and S_6) where S_1 and S_6 are respectively the SWS and TWS; and p_1 and p_2 are the transition probabilities.

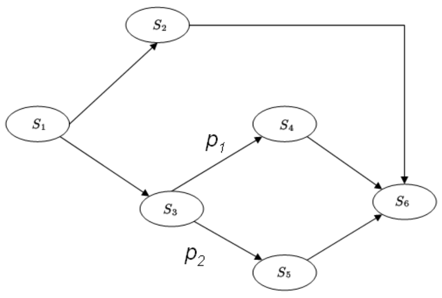

Figure 4. An example of composition graph

V. Physical Composition

The architecture of physical composition layer is given in Figure 5. The physical composition operation is composed of four steps. First, the Matchmaking Module uses the composition graph G to match each service type to the corresponding service instances. Second, Preliminary Analyzer Module scan the different instances and eliminates the ones that fail to meet the hard QoS constraints. Third, QoS Evaluator Module computes the global QoS for each (non eliminated) instance using the soft QoS attributes. Fourth, the Graph Transformer Module uses a set of mapping rules to construct a new composition graph representing the set of potential executable plans J.

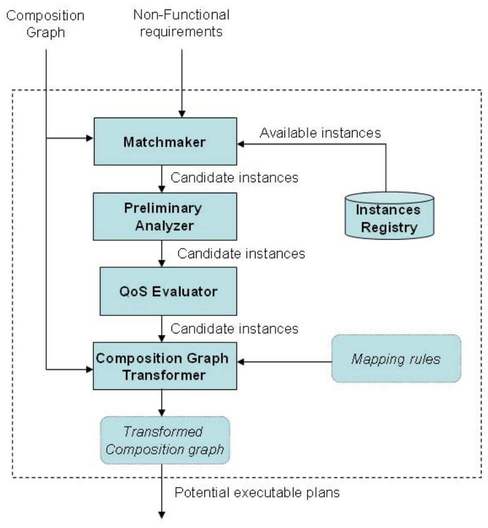

Figure 5. Architecture of physical composition layer

A. Identification of Candidate Instances

The first step in the physical composition is to identify, for each service type in the composition graph, the set of the corresponding instances. For this purpose, Matchmaker Module uses the composition graph and queries the service instances registry to associate to each service type a set of potential instances (at last one instance per type is needed). Note also that matchmaking in this level is based only on NFC requirements.

B. Preliminary Analysis

The role of the Preliminary Analyzer is to eliminate instances that fail to meet the constraints issued from the QoS attributes. Constraints are often implemented through elementary evaluation methods. Two commonly used methods are described below. Let $q_j(u)$ be the evaluation of instance u in respect to QoS attribute q_j.

- *Conjunctive method.* A minimal (resp. maximal) satisfaction level $\hat{q_j}$ is defined for each attribute $q_j \in Q^h$. An instance u is acceptable if and only if $q_j(u) \geq \hat{q_j}$ (resp. $q_j(u) \leq \hat{q_j}$) for all $j \in Q^h$.
- *Disjunctive method.* This method is similar to the previous one but an instance is considered as acceptable once at least one of its evaluations verifies the corresponding satisfaction level. An instance u is acceptable if and only if $\exists q_j \in Q^h$ such that $q_j(u) \geq \hat{q_j}$ for attributes with minimal satisfaction level or $q_j(u) \leq \hat{q_j}$ for attributes with maximal satisfaction level.

In addition to constraints, the preliminary analysis step implies also the use of the *dominance relation* to eliminate dominated instances. In contrary to constraints, which consider hard QoS attributes and apply indifferently to all the instance whatever the service type to which they are associated, the dominance analysis (i) is based on soft attributes only, and (ii) requires to consider separately the instances of each service type. The dominance relation in respect to a single attribute is defined as follows.

Definition 4: Let $q_j \in Q^s$ and let u and u' be two service instances. Then, instance u *dominates* instance u' in respect to attribute q_j, denoted $u\Delta_j u'$, if and only if: $q_j(u) \geq q_j(u')$.$\Diamond$

The dominance relation in respect to all soft attributes is defined as follows.

Definition 5: Let u and u' be two instances. Then, instance u *dominates* instance u' in respect to all soft attributes, denoted $u\Delta u'$, if and only if: $q_j(u) \geq q_j(u'), \forall j \in Q^s$ with a lest one strict inequality.$\Diamond$

In the two last definitions we assumed that attributes are to be maximized. However, the operator "$\geq$" should replaced by the operator "$\leq$" for attributes that should be minimized. Obviously, all denominated instances of a given service type should be eliminated.

C. QoS Evaluation of Service Instances

The objective of this step is to evaluate the QoS of all service instances. The formal model used to specify these evaluations is grounded on multicriteria evaluation. Let Ψ denotes a multicriteria classification model. As output of this step, each service instance u is assigned a QoS level $\Psi(u)$ on an ordinal scale $\mathcal{E}$ composed of a finite set of p evaluation levels: $\varepsilon_1 \prec \varepsilon_2 \prec \cdots \prec \varepsilon_p$.

Generally, the levels of scale $\mathcal{E}$ are defined in terms of profile limits $b_1, \cdots, b_{p-1}$ representing the boundaries between the different levels. Figure 6 shows the definition of three-level scale in terms of profile limits b_1 and b_2.

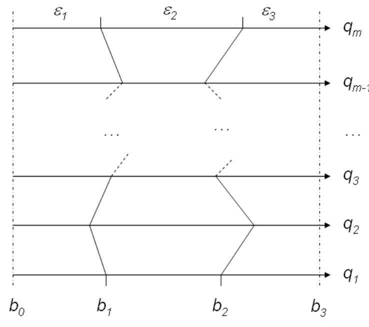

Figure 6. Definition of three-levels scale in terms of the profile limits

The computing of global QoS levels is formalized in Algorithm 1. The latter compares each service instance u to each of the profile limits staring from the highest one and assign to u the first QoS level for which u verifies the *assignment rule* associated with the lower profile limit of this level. The function *AssignmentRule* in Algorithm 1 corresponds to the assignment rule associated with Ψ. Algorithm 1 runs in $O(\alpha \times p)$, where $\alpha = |U|$, U set of service instances and p is the number of levels.

Algorithm 1: GlobalQoSLevel

```
Input  : $U$, // service instances.
         $\Psi$, // multicriteria classification model.
         $\mathcal{E}$, // evaluation scale.
Output: $\Psi(u), \forall u \in U$// global QoS levels.
$p \longleftarrow$number of levels in $\mathcal{E}$;
for ($all\ u \in U$) do
    $h \longleftarrow p$;
    assigned $\longleftarrow$ False;
    while ($h \geq 0 \wedge$ NOT(assigned)) do
        if (AssignmentRule$(u, h)$) then
            $\Psi(u) \longleftarrow h + 1$;
            assigned $\longleftarrow$ true;
        end
        $h \longleftarrow h - 1$;
    end
end
```

Different multicriteria classification models could be used. For instance, the assignment rule in the multicriteria classification model ELECTRE TRI [14] requires the computing of credibility indexes $\sigma(u, b_h) \in [0, 1]$. Then, an instance u is assigned to level h if and only if $\sigma(u, b_h) \geq \lambda$ where $\lambda \in [0.5, 1]$ is the *cutting level* parameter.

D. Composition Graph Transformation

Let $G = (S, R)$ be a generic composition graph. Then, G needs to be mapped to a new graph $H = (X, E)$ using different mapping rules. These rules differ along with the type of BPEL constructor such that probabilistic invocation, parallel invocation, sequential activation, fastest-predecessor-triggered activation, synchronized invocation and conditional invocation. The basic idea of the mapping rules consists in replacing each node $S_i \in S$ by three node types: one input node, one output node and several nodes corresponding to the instances associated with node S_i. For the purpose of illustration, we present the mapping rule relative to probabilistic invocation. The other mapping rules are defined in a similar way. First, we introduce some additional notations. Let:

- $\Gamma^+(S_i)$ be the set of successor nodes of S_i;
- ν be a neutral level on scale $\mathcal{E}$; and
- Φ and Ω be two aggregation operators.

Let now present the transformation rule for probabilistic invocation. In the probabilistic invocation, a probability value p_k on an outgoing arrow from S_i to S_j indicates that S_i invokes S_j with probability p_k. Let S_i be a node in S representing a service type with a probabilistic invocation. Let, for $k = 1$ to t, p_k be the invocation probability associated with edge $(S_i, S_k) \in R$ (see Figure 7.a). Then, the following mapping rule is applied:

Mapping rule 1.

```
1) For node $S_i$:
   • add to $X$ two nodes $S_{i,in}$ and $S_{i,out}$ with
     evaluation vector $(0, \nu)$
   • for each instance $u$ of $S_i$:
     – add to $X$ the node $u$ with evaluation
       vector $(C_i(u), \Psi(u))$;
     – add to $E$ the edges $(S_{i,in}, u)$ and $(u, S_{i,out})$;
2) For each node $A_k$ in $\Gamma^+(S_i)$:
   • add to $X$ the node $A_{k,in}$ with evaluation
     vector $(\Phi(A_{k,in}), \Omega(A_{k,in}))$;
   • add to $X$ the node $A_{k,out}$ with evaluation
     vector $(0, \nu)$;
   • for each instance $u$ of $A_k$:
     – add to $X$ the node $u$ with evaluation
       vector $(p_k \cdot C_i(u), \Psi(u))$;
     – add to $E$ the edges $(A_{k,in}, u)$ and
       $(u, A_{k,out})$;
3) For each node $A_k$ in $\Gamma^+(S_i)$:
   • add to $E$ the edge $(S_{i,out}, A_{k,in})$.
```

Mapping rule 1 is illustrated graphically in Figure 7. Let $h = 1, \cdots, t$. The operator Φ implies nodes on different levels and vary according to the BPEL constructors associated with each node. It may be the sum, product, max, min, or average. The operator Ω involves nodes on the same level and may be any aggregation operator such as sum, product, max, min, average, etc. Table I provides the formula for four basic web service evaluation attributes and for four basic BPEL constructors. More details for the definition of aggregation operators Φ and Ω are given in [8].

The evaluation vector $(\Phi(A_{h,in}), \Omega(A_{h,in}))$ associated with input node $A_{h,in}$ in Figure 7 is defined as follows: (i) $\Phi(A_{h,in})$ is an aggregation of the costs $C_i(A_1^1), \cdots, C_i(A_{n_h}^1)$ of service A_h instances; and (ii) $\Omega(A_{h,in})$ is an aggregation of the global QoS levels $\Psi(A_h^1), \cdots, \Psi(A_{n_h}^1)$ of service A_h instances.

Finally, it is important to note that when the attribute is ordinal (e.g. security), it is not possible to use the probability associated with the branches of *switch* constructor. Some solutions to avoid this problem are given in [8].

VI. Composition Evaluation and Selection

The architecture of evaluation and selection layer is given in Figure 8. It takes as input the transformed graph $H = (X, E)$ and generates a set of optimal executable plans from which the user should select one for effective execution. Evaluation and selection operation contains three steps: (i) transformation of graph H, (ii) resolution and identification of optimal executable plans, and (iii) selection of the executable plan to run.

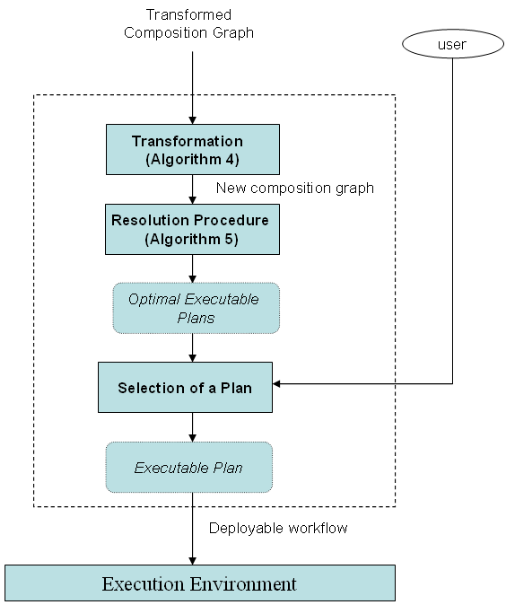

Figure 8. Architecture of evaluation and selection layer

A. Modeling of Potential Executable Plans

A composite web service can be seen as a sequence of invocation operations (i.e. edges) involving different individual web services (i.e. nodes). Therefore, a composite web service can be defined as a shortest path in graph H. More formally, a composite web service is defined as follows.

Definition 6: Let $H = (X, E)$ be a transformed composition graph with S_1 as SWS node and S_n as TWS node. A **composite web service** K is defined as a $S_1 - S_n$ path in H.$\Diamond$

Furthermore, two types of evaluations are considered:

- The qualitative evaluation $\Psi(S_i)$ associated with each vertex $S_i \in X$. These evaluations are meant to be used in a MinMax criterion to evaluate the composite Web service.
- The $C_i(S_i)$ cost associated with each vertex $S_i \in X$.

Table I
AGGREGATION OPERATORS FOR DIFFERENT ATTRIBUTES AND BPEL CONSTRUCTORS

Attribute	Sequence	Flow	Pick	Switch
Cost	$\sum_{k\in\Gamma^+(x)} q_j(k)$	$\sum_{k\in\Gamma^+(x)} q_j(k)$	$\sum_{k\in\Gamma^+(x)} q_j(k)$	$\sum_{k\in\Gamma^+(x)} \alpha_k \cdot q_j(k)$
Response time	$\max_{k\in\Gamma^+(x)} q_j(k)$	$\max_{k\in\Gamma^+(x)} q_j(k)$	$\sum_{k\in\Gamma^+(x)} q_j(k)$	$\sum_{k\in\Gamma^+(x)} \alpha_k(x,y) \cdot q_j(k)$
Availability	$\prod_{k\in\Gamma^+(x)} q_j(k)$	$\prod_{k\in\Gamma^+(x)} q_j(k)$	$\sum_{k\in\Gamma^+(x)} q_j(k)$	$\sum_{k\in\Gamma^+(x)} \alpha_k \cdot q_j(k)$
Security	$\min_{k\in\Gamma^+(x)} q_j(k)$	$\min_{k\in\Gamma^+(x)} q_j(k)$	by rule	by rule

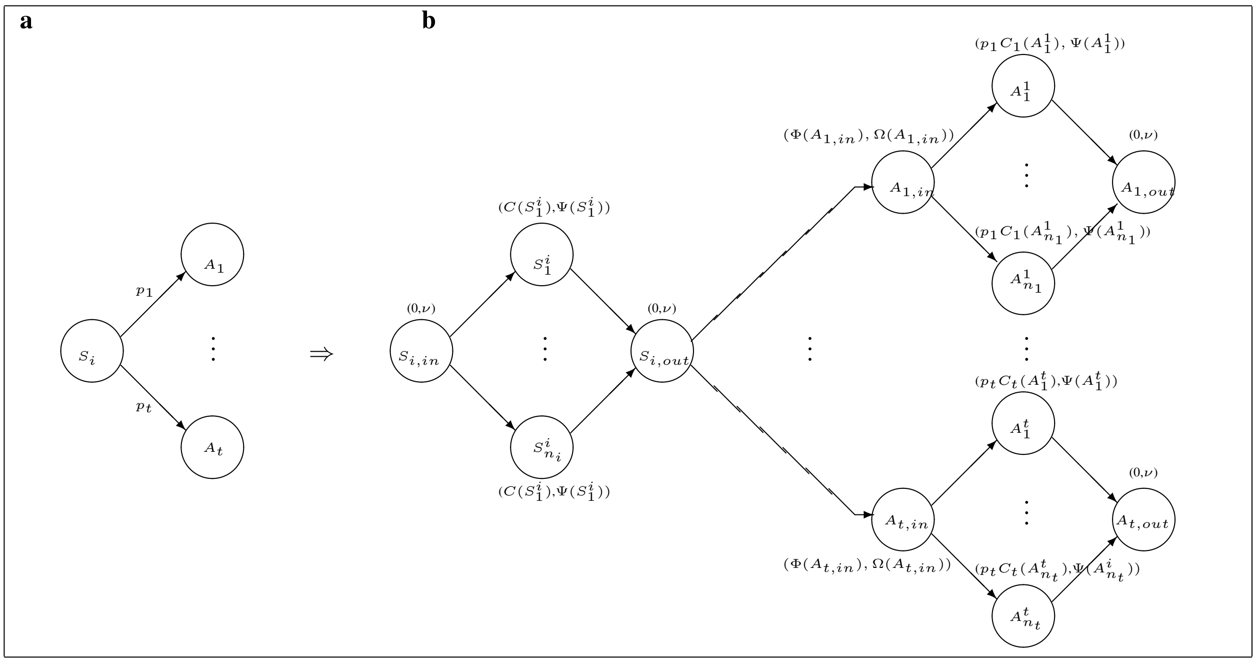

Figure 7. Probabilistic invocation transformation

B. Transformation of Graph H

As mentioned earlier, the selection of the optimal executable plans is based on the resolution of a bi-objective shortest path problem. This requires first to transform the graph H into a new graph H' as follows (see Figure 9). For each edge (S_i, S_j) we consider a vector of two evaluations $e(S_i, S_j) = (C_{ij}(S_i, S_j), \tau(S_i, S_j))$ where $C_{ij}(S_i, S_j)$ corresponds to the cost for invoking service instance S_j from S_i and $\tau(S_i, S_j) = \max\{\Psi(S_i), \Psi(S_j)\}$.

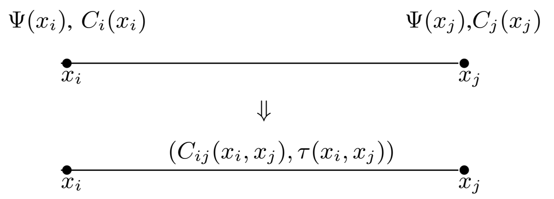

Figure 9. Transformation schema

The reason for defining $\tau(S_i, S_j)$ in this way is related to the fact that $\Psi(.)$ is considered as a MinMax criterion. In fact, any path which includes the edge (S_i, S_j) should take into account for both $\Psi(S_i)$ and $\Psi(S_j)$; hence the maximum. The construction of graph H' according to this transformation schema is formalized in Algorithm 2, which runs in $O(n)$ where n is the cardinality of X.

Algorithm 2: GraphHTransformation

```
Input  : $H = (X, E)$, // transformed composition graph.
Output: $H' = (X', E')$, // new transformed composition
        graph.
$X' \longleftarrow X$;
$E' \longleftarrow \emptyset$;
for $(all\ (u, v) \in E)$ do
    $e_1 \longleftarrow c(v)$;
    $e_2 \longleftarrow \max\{\Psi(u), \Psi(v)\}$;
    $E' \longleftarrow E' \cup (u, v)$; with $e(u, v) = (e_1, e_2)$;
end
$H' \longleftarrow (X', E')$;
return $H'$
```

C. Identification of Optimal Executable Plans

In order to solve this problem, we will use the algorithm that we proposed in [15]. The idea of this algorithm is as follows. Since each edge (S_i, S_j) in the transformed graph H' is evaluated by vector $e = (C_{ij}(S_i, S_j), \tau(S_i, S_j))$, the complexity of the computation of the efficient set can be strongly reduced by solving a number of mono-objective shortest path problems only. The efficient set is constructed by solving the following bottleneck shortest path problems: find if there is a shortest path $\mathcal{P}$ from s to t using cost $C_{ij}(S_i, S_j)$ such that $\max_{(S_i,S_j)\in\mathcal{P}} \tau(S_i, S_j) \leqslant \nu$, for $\nu = 1, \dots, p$. To solve each of these problems, edge (S_i, S_j) such that $\tau(S_i, S_j) > \nu$ are deleted from the graph and the classical Dijkstra's algorithm is applied.

This resolution procedure is summed up in Algorithm 3 where function *Dijkstra* implements the classical Dijkstra's algorithm and returns the shortest path of level ν in graph $H'_\nu = (X_\nu, E_\nu)$. The complexity of the Algorithm 3 is $pD(r, n)$, where $r = |E'|$ and $D(r, n)$ is the complexity of computing a shortest path.

Algorithm 3: BestCompositions

```
Input  : $H' = (X', E')$, // new transformed composition graph.
         $\mathcal{E}$, // evaluation scale.
Output: $\mathfrak{P}$, // best compositions.
$p \longleftarrow$ number of levels in $\mathcal{E}$;
$\nu \longleftarrow 1$;
$\mathfrak{P} \longleftarrow \emptyset$;
while ( $\nu \leq p$) do
    $X'_\nu \longleftarrow \{S_i \in X' : \exists S_j \in X' \wedge \tau(S_i, S_j) \leq \nu\}$;
    $E'_\nu \longleftarrow E' \setminus \{(S_i, S_j) \in E' : \tau(S_i, S_j) > \nu\}$;
    $H'_\nu \longleftarrow (X'_\nu, E'_\nu)$;
    $P_\nu \longleftarrow$ Dijkstra($X'_\nu, E'_\nu$);
    $\mathfrak{P} \longleftarrow \mathfrak{P} \cup \mathcal{P}$;
    $\nu \longleftarrow \nu + 1$;
end
return $\mathfrak{P}$
```

D. Selection the Executable Plan to Run

The resolution procedure generates at most p executable plans where p is the number of levels in $\mathcal{E}$. These plans are ordered along with their tradeoff of the cost and the global QoS attributes. From this reduced set of composite web services, the user should select only one plan to deploy.

VII. Conclusion

We proposed a layered system for web service composition. The proposed system addresses the problem of web service composition from end to end perspective, i.e. from end-user specification of the desired service to service deployment. The input for the system are the specifications of the desired service. It outputs a reduced set of executable plans that tradeoff at best the cost and the global QoS attributes. The system is currently being implemented. Implementation details will be the subject of a forthcoming paper.

References

[1] S. Hashemian and F. Mavaddat, "A graph-based approach to Web services composition," in *Proceedings of the International Symposium on Applications and the Internet (SAINT 2005)*, Trento, Italy, 31 Jan. - 4 Feb. 2005, pp. 183–189.

[2] E. Maximilien and M. Singh, "A framework and ontology for dynamic Web services selection," *IEEE Internet Computing*, vol. 8, no. 5, pp. 84–93, 2004.

[3] F. Siala and K. Ghedira, "How to select dynamically a QoS-driven composite Web service by a multi-agent system using CBR method," *International Journal of Wireless and Mobile Computing*, 2012, (*in press*).

[4] Z.-P. Gao, J. Chen, X.-S. Qiu, and L.-M. Meng, "QoE/QoS driven simulated annealing-based genetic algorithm for Web services selection," *The Journal of China Universities of Posts and Telecommunications*, vol. 16, no. Supplement 1, pp. 102–107, 2009.

[5] R. Krithiga, "QoS-aware web service selection using SOMA," *Global Journal of Computer Science and Technology*, vol. 12, no. 10, pp. 47–51, 2012.

[6] L. Zeng, B. Bentallah, M. Dumas, J. Kalagnanam, and Q. Sheng, "Quality driven web service composition," in *Proceedings of the 12th international conference on World Wide Web*, Budapest, Hungray, 20-24 May 2003, pp. 411–421.

[7] F. Casati, S. Ilnicki, L.-J. Jin, V. Krishnamoorthy, and M.-C. Shan, "eFlow: a platform for developing and managing composite e-services," HP Laboratories, Palo Alto, Technical Report HPL-2000-36, 2000.

[8] S. Chakhar, S. Youcef, V. Mousseau, L. Mokdad, and S. Haddad, "Multicriteria evaluation-based conceptual framework for composite Web service selection," Lamsade, University Paris Dauphine, France, Research Report, 2011.

[9] V. Chifu, C. Pop, I. Salomie, M. Dinsoreanu, A. Niculici, and D. Suia, "Selecting the optimal web service composition based on a multi-criteria bee-inspired method," in *Proceedings of the 12th International Conference on Information Integration and Web-based Applications and Services (iiWAS)*, Paris, France, 8-10 Nov. 2010, pp. 40–47.

[10] D. Menascé, "Composing Web services: A QoS view," *IEEE Internet Computing*, vol. 6, no. 8, pp. 88–90, 2004.

[11] D. Menascé and V. Dubey, "Utility-based QoS brokering in service oriented architectures," in *Proceedings of the International Conference on Web Services (ICWS)*, Salt Lake City, Utah, USA, 9-13 July 2007, pp. 422–430.

[12] D. Petrova-Antonova and A. Dimov, "Towards a taxonomy of web service composition approaches," *Scalable Computing: Practice and Experience*, vol. 12, no. 4, pp. 377–384, 2011.

[13] V. Agarwal, G. Chafle, K. Dasgupta, N. Karnik, A. Kumar, S. Mittal, and B. Srivastava, "Synthy: A system for end to end composition of web services," *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 3, pp. 311–339, 2005.

[14] J. Figueira, V. Mousseau, and B. Roy, "Electre methods," in *Multiple criteria decision analysis: State of the art surveys*, J. Figueira, S. Greco, and M. Ehrgott, Eds. Springer-Verlag, New York, 2005, pp. 133–162.

[15] H. Aissi, S. Chakhar, and V. Mousseau, "GIS-based multicriteria evaluation approach for corridor siting," *Environment and Planning B: Planning and Design*, vol. 39, no. 2, pp. 287–307, 2012.