Abstract—Web service selection involves finding services from a possibly large database of similar services. Hence, the challenges involved in finding a suitable service include large time consumption, and difficulty of finding a perfect match according to the user specified search keywords. For instance, users may have privacy and security concerns, as the information involved with service selection and provisioning may be sensitive for both providers and users.

In this paper, we define an approach to provide customized recommendation of composite services according to a variety of user-specified criteria, including classic quality of service as well as complex privacy and security dimensions. We conduct an extensive experimental evaluation, using datasets of actual WSDL documents and comparing our algorithms with state-of-the-art solutions. Our experimental evaluation demonstrates that our algorithms are both effective and efficient.

I. INTRODUCTION

The growing success of Web services as the preferred standards-based approach to implement distributed and heterogeneous systems has resulted in a large number of providers offering services of varying degree of sophistication and complexity, ranging from weather forecast to travel management. On one hand, the availability of a wide array of services has created a competitive and flexible market that meets the needs of different type of users (e.g. individuals, organizations, law enforcement etc.). On the other hand, this large availability of heterogeneous services makes the service selection process a non-trivial task, making Web service selection and provisioning play a crucial role in Web service life-cycle. Here, several application-dependent requirements might constrain the selection of the best service.

Existing literature offers some approaches toward an efficient and effective selection of Web services [1], [5], [11], [16], [18]. However, although all existing efforts acknowledge the need of assisting users in identifying the best available services, current recommenders for Web Services are primarily tailored for "standardized" ranking based on Quality of Service criteria (QoS) [1], [4], rather than personalized users' criteria. We notice that while QoS criteria are extremely important, Web services may be invoked by a diversified population of users, who may be more or less technically savvy, and therefore express criteria of various kind and granularity, beyond QoS. For example, users might be interested in filtering Web services with inaccurate vendor implementations, or coding mistakes that could lead to exploitable security vulnerabilities. Alternatively, they may request privacy policy compliance, or preferred quality of service. This points out the need of a recommendation system for Web services able to accommodate multiple users’ search criteria on dimensions which are highly disparate.

In order to accommodate users’ needs, these kinds of criteria should be expressed at varying degrees of granularity/level, and allow users to pose constraints on the desired value for a given dimension, e.g., the degree of security level, as well as express specific conditions on the Web services properties, e.g., the desired encryption algorithm. The criteria should be considered for evaluation, according to their importance (as indicated by the users) and the extent to which a satisfactory instance or a composition of Web services can be found. To guarantee flexibility, the users’ criteria should be considered even if they cannot be fully met by the candidate Web service providers, and used to the extent possible to identify the best available services.

Motivated by the above considerations, the goal of this work is to design and develop a reliable and user-centric recommendations mechanism of single and composite Web services, where the recommended services are selected according to user-entered criteria. We design and develop WS-Rec, which provides recommendation of single and composite Web-services taking into account users’ criteria in an accurate and yet efficient fashion. The criteria may be expressed at various levels of granularity, belonging to multiple domains (privacy, quality of service, security etc.) and be ranked by the user in the order of importance. In order to compute recommendations, WS-Rec implements various strategies, customized by the users’ input and preferences. Our algorithms support cases where users enter some criteria, and also can operate according to any specified ranking preference (i.e. rank the criteria for selection), or if a minimum acceptable level for any simple or aggregated criteria is provided. Our strategies are modeled through classic multi-criteria optimization problems, thus ensuring low complexity and tractability. Moreover, we conduct experimental evaluation using the state-of-the-art real-world data sets. As demonstrated by our empirical evaluation, our algorithms are efficient with respect to both time and computation.

The paper is organized as follows. In Section II, we provide an overview of the relevant literature in the area. In Section III, we describe our proposed framework. We present the language used for representation of rules in Section IV. In Section V, we present our ranking model for service delivery. In Section VI, we report some experimental evaluation. We conclude in Section VII.
II. RELATED WORKS

Sellami et al [16] propose an architecture that recommends service registries that have the highest probability to satisfy the user’s query. In recommending registries to the user, they take into consideration the user’s domains of interest, invocation history, and non-functional requirements. Similar to Sellami et al’s work, Liu et al [14] propose a Web service discovery approach by grouping similar services together and then dispatching the consumer requests to suitable services in terms of QoS requirements. Kalepu et al [11] use QoS criteria to predict the trustworthiness of the service providers by computing the difference between the QoS values portrayed by the services and the actual QoS values. These works, including [5], [16], [11], focus on selection of a single service and do not support recommendations for composite services.

Composition based on QoS criteria has been studied by several authors, e.g. [21], [1], [19], [20]. Claro et al [9] propose a recommendation system for Web service compositions that utilizes the Gaussian Bayes Classifier to predict the combinations of services suitable to the user profile. This approach takes user ratings as input. As it will be discussed in Section IV, our input to the recommender system is user preferred criteria. Zeng et al [20] present a middleware aimed at selecting Web services for composition such that user satisfaction is maximized when it is expressed as a utility over QoS functions. As it will be presented in Section V, we consider how each of the user criteria affect the ranking of the service combinations, in contrast to Zeng’s work. Also, in [6], [13], [20], the authors build a suitable composition by taking into consideration all possible candidate services for each activity in a work flow. In this work, we assume that all possible service combinations in the form of work flows are provided to our framework, and we identify and rank the best service combinations which when executed will lead to a composition. Finally, approaches for service selection based on fuzzy-logic have been recently proposed such as [10], [3]. Almulla et al [3] suggest non-exact matching techniques to rank available Web services. In addition to employing non-exact matching techniques, we aim at providing: 1) a customized ranking protocol, which strictly adheres to the user preferences of different granularity and types, 2) an expressive language for users to define user-criteria, which can be matching conditions, or high-level criteria, 3) various strategies to accommodate users’ input, and 4) a solution to deal with complex scenarios of service compositions, rather than single services.

III. OVERALL ARCHITECTURE OF THE WS-REC

The WS-Rec acts as a broker on behalf of the user, and is in charge of helping users select the best available services for their need. In order to do so, WS-Rec takes two types of input. First is a description of the service, defined by service type, input, output and possible functional constraints. Second type of input is the set of user criteria, defined over non-functional properties (or capabilities). For example, user criteria can relate to quality of service, privacy & security requirements.

For each criteria, the user can express one or multiple fine-grained conditions (e.g., time of execution lower than 2 hours, maximizing overall QoS), and the user can provide an order of preference or ranks for the criteria.

Using the service description, as a first step, WS-Rec finds the workflow of activities which are able to produce the desired outputs.1 In support of this task, we assume the availability of libraries associated with each service description, and a set of possible standard business processes and the corresponding WFs generated by a planner module [9]. A planner module generates the work flow of activities by taking the description, and input and output information from the user (see [9] for a detailed description). For instance, the user provides the service description for a travel service as airline and hotel reservation, with input being date of travel, and source and destination locations, and output being the cost of the trip. The activities generated by the planner module would be airline reservation, hotel reservation and payment service.

Then, WS-Rec according to the activities obtained from the planner module, queries WS public registries (e.g., UDDI) to retrieve the set of candidate services able to carry out the corresponding WF activities. We assume that each service description (e.g., WSDL file) is annotated with the semantic and syntactic information needed to match the user criteria with. The WSDL file of a service can be annotated with both functional as well as non functional capabilities of a service as described in [7], [15], [12]. Upon locating the candidate services and computing candidate compositions meeting the functional requirements, WS-Rec executes the multi-criteria recommendation algorithms. First, WS-Rec constructs a directed acyclic graph (DAG) representing all identified service compositions. It then evaluates users’ search criteria, so as to add weights in the graph, wherein each weight represents the degree of satisfaction of the user criteria by each candidate service. The degree of satisfaction may be 100%, or lower, depending on whether users’ requests are fully met. Next, WS-Rec computes the best service compositions, according to a recommendation strategy. Here, the best service compositions are the compositions (among those in the DAG) that maximize the satisfaction of the user’s preferences. Precisely, the algorithm returns a ranked list of suitable service compositions (or single services, if no compositions were needed), taking into account imprecise or vaguely defined conditions, as well as any fine-grained conditions the user may have specified. This list is returned to the user, who can at this point make a final selection of a combination and request the Business Process Execution Language (BPEL) engine to instantiate the composition.

**Example 1:** Consider a user requesting a travel service which includes airline reservation ($SP_1$), hotel reservation ($SP_2$) and payment services ($SP_3$). As such, the travel service is a composite service with three types of services. The user provides the search interface, the description of the type of

1Note that the user request could also imply a single service invocation. In our paper, this is modeled as a workflow with a unique activity.
service he or she desires, that is, a travel service, along with input as the dates of travel, and source and destination locations as part of functional requirements. Additionally, the user provides the following non-functional requirements: transaction to occur over a secure and encrypted channel and execution time to be less than 10 seconds. Ideally, the user would like to be informed about the possible Web services options available to him. Once a suitable service or combination of services is provided, the user will be able to select a suitable service (or combination of) that best meet its requirements.

IV. USER SEARCH CRITERIA AND SERVICE PROVISIONING RULES

User search criteria are expressed against non-functional properties of the Web service, and can be represented as boolean expressions of atomic conditions, which are matched against Web service descriptions.\(^2\) In the following, we first formally introduce the atomic condition, and user search criteria. We then introduce the concept of satisfaction level, so as to measure the degree to which a user criteria is unmatched.

A. Criteria and Rule conditions

Each Web service is associated with a set of non-functional properties, such as estimated time of execution, adopted algorithm for data encryption, retention time of personal data, cost of execution, etc. We assume that all such properties are organized into taxonomies, based on their common domain. For instance, \textit{privacy} (e.g., retention time, third-party usage, purpose), \textit{security} (e.g., encryption algorithm, signature algorithm), and conventional \textit{QoS} (e.g., response time, availability, throughput), represent taxonomies. According to this representation, users search criteria are represented as boolean expressions combining \textit{atomic conditions} on Web service properties. Atomic conditions can be expressed in two non-exclusive forms: \textit{low-level conditions}, i.e., conditions matched against individual Web service properties, and \textit{high-level conditions} which enable the user to indicate the taxonomy whose service properties he or she wishes to maximize or minimize, without stating specific conditions on low level properties. For instance, a user may indicate that he wishes his privacy to be maximized, without providing conditions on the properties representing typical components of services’ privacy policies (e.g., retention, purpose, etc.). We formally define atomic conditions as follows.

\textbf{Definition 1 (Atomic condition):} Let \( P \) be the set of properties, \( T \) be the set of taxonomies defined on \( P \). An atomic condition \( \text{cond} \) can be in three forms: \text{CAP} \( \text{OP} \) \text{value}, \text{Max(CAP)}\), and \text{Min(CAP)}\) respectively, where: (1) \( \text{CAP} \in P \) is a property, (2) \( \text{OP} \in \{=,e,\geq,>,\leq,<,\subseteq,\supseteq,\supset,\subseteq,\inf\} \) is a matching operator; (3) \text{value} is the user preferred value for \text{CAP}.

\(^2\)Notice that we limit our analysis to criteria that can be evaluated without having to execute the composition.

According to the above definition, we support two versions for a same operator, \( OP_e, OP_p \) (e.g., exact equality \( = \) and partial equality \( =p \)). With \( OP_e \) (\( OP \) denotes any boolean operator per Definition 1) we denote a condition that has to be matched exactly to be considered satisfied. That is, \( \text{time} <_{e} 10\text{hrs} \) denotes the request that only services with execution hours with less than one hour are considered. Conversely, \( p \) denotes an imprecise condition, wherein a service would be considered for recommendation even if the condition is not matched (but a degree of satisfaction can be measured, as discussed next). Note also that for the equality operator we give two different versions on the loose matching, denoted as \( =_{inf} \) and \( =_{sup} \), as this operator has possibly two different semantics. Indeed, a user might require that a specific property should have a given value/set of values and that it is not admissible having a superset or a subset of the required set (denoted with the \( =_{inf} \) and \( =_{sup} \) operator, respectively).

A user search criteria is defined as follows.

\textbf{Definition 2 (User Search Criteria):} A user search criteria \( uc \) is represented as a pair \((exp, pr)\) where \( exp \) is a boolean expression of atomic conditions (see Definition 1), whereas \( pr \in [0,1] \) represents the user-specified priority level, associated with the user criteria.

Users may specify one or more search criteria. Depending on their scope of application, search criteria may be \textit{global} or \textit{local}. Local criteria apply to individual services in a given composition, whereas global criteria can only be evaluated against the overall composition as a whole.

\textit{Example 2:} In the travel reservation scenario, two local user criteria are:

\( uc_{1} \equiv (\{\text{ExecPrice}<_{p} 500, 0.3\} \text{AND} (\text{CryptoAlgo}=_{sup} \{\text{TDES}, \text{AES}\}, 0.7)) \),

where 0.3 and 0.7 are the priority levels of the user criteria respectively. A global criteria is \( \text{ExecTime} <_{e} 3000\text{ms} \), meaning that only services with execution time bounded to 3000 ms are considered.

B. Satisfiability of conditions

We introduce the notion of satisfaction level for user search criteria. This aims at quantifying the distance between the property requested and property guaranteed by the service. This value ranges in the interval \([0,1]\), where 1 represents exact matching and 0 represents no matching at all. We begin with showing how to compute the satisfaction of a single condition. As shown, the exact computation is functional to the semantic of the operator in the condition.

\textbf{Definition 3 (Satisfaction of Atomic Condition):} Let \( Cond \) be an atomic condition, and \( v \) be the value of property \text{CAP} for Web service \( WS \). The satisfaction level of \( WS \ w.r.t \ Cond \), denoted as \( sat_{WS}(Cond) \), is computed according to the rules in Table I. In the table, given a property \text{CAP} \( \in C \), we assume that that \( Dom(\text{CAP}) \) is an ordered set denoting all possible input values (e.g., \( Dom(\text{CAP}) \subset \mathbb{N} \)), and that \( MinDom()\text{MaxDom()} \) return the minimum/maximum value in this domain set.

\textit{Example 3:} Consider the following user criteria:

\( \text{RetTime} =_{p} 30 \text{ AND AllowedInfo}=_{inf}\{\text{ContactInfo},\text{ContactNumber}\} \)
<table>
<thead>
<tr>
<th>OP</th>
<th>Value’s domain</th>
<th>Satisfaction level ((Sat_{\omega}(\text{Cond})))</th>
<th>Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\leq), (\geq), (\approx)</td>
<td>(\in \text{Dom}(\text{CAP}))</td>
<td>1</td>
<td>\textbf{Cond} is satisfied</td>
</tr>
<tr>
<td>(\in_{\text{inf}} (\rlap{&quot;\text{inf}&quot;})) (\in_{\text{sup}} (\rlap{&quot;\text{sup}&quot;}))</td>
<td>value, (v \in \text{Dom}(\text{CAP}))</td>
<td>0</td>
<td>\textbf{Cond} is not satisfied</td>
</tr>
<tr>
<td>(\leq) (\cap) (\geq)</td>
<td>value, (v \in \text{Dom}(\text{CAP}))</td>
<td>((1 - \frac{</td>
<td>\text{value} \cap v</td>
</tr>
<tr>
<td>(\leq) (\cup) (\geq)</td>
<td>value, (v \in \text{Dom}(\text{CAP}))</td>
<td>((1 - \frac{</td>
<td>\text{value} \cup v</td>
</tr>
<tr>
<td>(\leq) (\lor) (\geq)</td>
<td>value, (v \in \text{Dom}(\text{CAP}))</td>
<td>((1 - \frac{</td>
<td>\text{value} \lor v</td>
</tr>
<tr>
<td>(\leq) (\land) (\geq)</td>
<td>value, (v \in \text{Dom}(\text{CAP}))</td>
<td>((1 - \frac{</td>
<td>\text{value} \land v</td>
</tr>
</tbody>
</table>

**TABLE I:** Partial satisfaction operations for conditions of the form \(\text{Att} \ OP \ \text{value}\), with \(v\) denoting the actual value of the attribute being matched.

demographicInfo \(\) AND \(\text{ATP}={\{\text{none}\}}\), and assume that the flight reservation Web service \((W.S_1)\) has the following properties: \(\text{AllowedInfo}={{\{\text{contactInfo}, \ \text{demographicInfo}, \ \text{financialInfo}\}}\) AND \(\text{RetTime}=40\) AND \(\text{ATP}={\{\text{WS2}\}}\). The corresponding levels of satisfaction are: \(s_{\text{Att}(W.S_1)}(\text{AllowedInfo}=\in_{\text{inf}}(\{\text{contactInfo}, \ \text{demographicInfo}\})=0;\)\(\)

\(s_{\text{Att}(W.S_1)}(\text{RetTime}=p \ 30)-(1 - \frac{|360-30|}{360}) = 0.973\), where \(\text{Dom}(\text{RetTime})=365\) days; \(s_{\text{Att}(W.S_1)}(\text{ATP}=\{\text{none}\})=0\). As such, \(s_{\text{Att}(W.S_1)}(\text{uc}) = 0\).

V. RANKING MODEL FOR WS DELIVERY

In this section, we introduce the data model over which WS-Rec computes the ranking of Web service compositions to provide users with customized recommendations, as well as the strategies to combine together for criteria combination.

A. Graph Model for Service Composition

Given a workflow \(W.F\), defined as a sequence of activities \(a_1, \ldots, a_n\), and the set of candidate Web services for each of them, we make use of a graph abstraction to account for all compositions resulting from all possible combinations of candidate Web services that satisfy \(W.F\) according to the functional requirements and compatibility of formalisms. In general, a level \(j\) of the graph is related to the \(j\)-th activity in the WF, where each node at level \(j\) represents a candidate Web service able to perform activity \(j\). Therefore, each path in the graph, referred to as \emph{composition graph}, denotes a potential composition. The candidate services are connected in the graph according to the functional capabilities of the candidate services. That is, the candidate services are connected only when their outputs and inputs are compatible with each other. Though, two services might perform the same task, the format of their input and output might be different. Hence, the two nodes corresponding to these two services are not necessarily connected to the same node. Hence, not all nodes in a level in the graph might be connected to all other nodes in a different level. As the recommendations are based on user criteria, that is, on whether and how much candidate Web services satisfy user criteria, the \emph{composition graph} is defined as a weighted graph [8]. More precisely, given the set of user criteria \(UC=\{uc_1, \ldots, uc_n\}\), the composition graph assigns to each edge, a vector of \(n\) elements corresponding to satisfaction levels of each local user criteria in \(UC\). Possible global criteria are not depicted in the graph, but considered in the strategy as need be. For clarity of presentation, we assume that one graph is created per workflow.

\emph{Definition 4 (WS-Graph):} Let \(W.F=\{a_1, \ldots, a_n\}\) be a sequentialized workflow of activities, \(W.S_{\text{cond}}\) be the set of candidate Web services associated with each \(a_j \in W.F\), and \(\text{UC} = \{uc_1, \ldots, uc_n\}\) be the set of user criteria. The Composition Graph \(G = < V, E, \Phi >\) is a weighted directed acyclic graph, where \(V\) is the set of vertices representing Web services in \(W.S_{\text{cond}}\); \(E\) is the set of edges, where \(\forall v, v' \in E\) iff \(v\) and \(v'\) are candidate Web services for activities \(a_i\) and \(a_{i+1}\) in WF, respectively. The labeling function \(\Phi\) is defined as \(\Phi; E \rightarrow < R^+, R^- >\), where for each \(\forall v, v' \in E\), \(\Phi(e)=< (\text{sat}_v(uc_1), \text{pr}(uc_1)), \ldots, (\text{sat}_v(uc_n), \text{pr}(uc_n)) >\), that is, \(\Phi\) returns the array whose components report for each
user criteria in \( u_c \) ∈ \( UC \) the satisfaction levels of \( v \) w.r.t. \( u_c \) and user priority weights for each criteria \( pr(u_c) \), where \( \sum_{j=1}^{\lambda} pr(u_c) = 1 \).

The labeled edges model the level of satisfaction of user’s criteria \( uc \) by the incoming node (Web service), along with the corresponding priority \( pr(uc) \). By traversing the graph’s paths, it is possible to find the most suitable Web service compositions meeting the local criteria entered by the user.

**Example 4:** With respect to Example 1, WS-Rec finds a set of 4 Web service combinations. These combinations are generated according to the generated workflow, that consists of the following three activities: airline reservation, hotel reservation, and payment. WS-Rec builds the composition graph represented in Figure 1 out of the 4 combinations, computes the satisfaction levels and assigns the priority weights provided by the user for each type of criteria. Accordingly, it assigns the pair of satisfaction value and user priority weight for each criteria to the edges. The graph has three levels, where each level consists of services with equivalent activities. In addition, the graph includes a source and destination node to the graph which are just dummy nodes acting as starting and ending nodes in the graph to enable easier path evaluation.

### B. Composite Web service ranking Strategies

WS-Rec supports three strategies to compute the recommendations: Average Criteria, Priority-based, and Global Evaluation. The strategies rely on the WS-Graph of Definition 4, and are elegantly reduced to the well-known problem of finding the set of shortest paths in the composition trees, with multi-criteria edges. This class of problems, referred to as multi-objective shortest path (MOSP) problems, can be solved in an optimal fashion with reasonable complexity [17]. In order to apply this class of solutions, we modify the original multi-objective shortest path problem methods to solve our need for multi-objective optimal path problem.

1) **Input on all criteria: Weighted Average of Criteria:** In this strategy, paths are ranked according to priority weights assigned by users for each criteria. To enforce this strategy we define a *path evaluating function*, \( PEF \), defined as the weighted average of the criteria values [17].

Let \( G \) be a composition graph computed over a set of user criteria \( UC \) according to Definition 5, where \( Set_P \) is the set of paths in \( G \). Moreover, let \( P \) nodes \( = \{P.v_1, P.v_2, ... P.v_{n-1}, P.v_n\} \) be the set of nodes (i.e., Web services) belonging to path \( P \), for each \( P \) ∈ \( Set_P \). We denote by \( P_{max} \) the path in \( Set_P \) with the highest sum of satisfaction levels for criteria \( u_c \). Given a \( u_c \) ∈ \( UC \), the function for calculating the average of satisfaction levels of \( u_c \) along path \( P \) is called \( AF_{uc}(P) \) and computed as follows:

\[
AF_{uc}(P) = \frac{\sum_{\forall v_j \in P \text{nodes}} sat_{v_j}(u_c)}{\sum_{\forall v_j \in P_{max} \text{nodes}} sat_{v_j}(u_c)}
\]

The path evaluation function is \( PEF(P) = \sum_{i=1}^{\lambda} pr(u_c) \times AF_{uc}(P) \). Finally, we rank the paths according to the assigned PEF values, obtaining \( R_{Path_{set}} \).

The higher the value assigned, the higher is the rank of the path or a set of Web services in the path.

**Example 5:** With respect to the graph of Figure 1, first, we compute the sum of satisfaction values for each criteria. Let \( P_1 \) be the path joining the nodes S, F1, H1, PS1, and D, \( P_2 \) be the path joining S, F1, H2, PS2, and D, \( P_3 \) be the path joining S, F2, H2, PS2, and D, and \( P_4 \) be the path joining S, F2, H3, PS2, and D. For the first criteria \( u_c \) and first path \( P_1 = \{P.1.S, P.1.F1, P.1.H1, P.1.PS1, P.1.D\} \) in Figure 1, the sum of satisfaction levels is computed as, \( \sum_{j=1}^{\lambda} |P_j \text{nodes}| sat_{v_j}(u_c) = 0.7 \). Similarly, for the next set of paths, \( \sum_{j=1}^{\lambda} |P_j \text{nodes}| sat_{v_j}(u_c) = 1.4 \), \( \sum_{j=1}^{\lambda} |P_j \text{nodes}| sat_{v_j}(u_c) = 1.5 \), and \( \sum_{j=1}^{\lambda} |P_j \text{nodes}| sat_{v_j}(u_c) = 1 \). Path \( P_3 \) has the highest sum of satisfaction levels. The average of satisfaction values for path \( P_1 \) and criteria \( u_c \) is computed as \( AF_{uc}(P_1) = \frac{\sum_{\forall v_j \in P \text{nodes}} sat_{v_j}(u_c)}{\sum_{\forall v_j \in P_{max} \text{nodes}} sat_{v_j}(u_c)} = 0.47 \). Similarly, for the next set of paths, \( AF_{uc}(P_2) = 0.93 \), \( AF_{uc}(P_3) = 1 \), and \( AF_{uc}(P_4) = 0.67 \). Similar computations are performed on the remaining criteria for all paths.

Finally, we compute the PEF value for each path whose values for each of the four paths are as follows: \( PEF(P_1) = 0.710 \), \( PEF(P_2) = 0.765 \), \( PEF(P_3) = 0.8575 \), and \( PEF(P_4) = 0.685 \). The PEF value of path 3 is the highest among the other paths. Hence we rank the paths in the following order from highest to lowest: \( \{P_3, P_2, P_1, P_4\} \).

2) **Priority-Based ranking:** This strategy is implemented when the user provides a single high priority criteria. The strategy attempts to maximize satisfaction level for that criteria, while checking for satisfaction of any other criteria the user may have entered. From an algorithmic standpoint, the high priority criteria is considered as the input of an *objective function*, according to the original formulation of the Hierarchization of Objective Functions algorithm [17]. Precisely, we adapt the original algorithm as follows. Let \( u_c \) be the user criteria with highest priority, and \( u_c[k] \) be lower priority criteria, sorted by importance. For each path \( P \) in \( Set_P \), we consider the sum of satisfaction levels with respect to criteria \( u_c \), obtaining a new PEF function:

\[
PEF(P) = \frac{\sum_{j=1}^{\lambda} sat_{v_j}(u_c)}{\sum_{j=1}^{\lambda} sat_{v_j}(u_c)}
\]

The paths in \( Set_P \) are then sorted according to the PEF function value, and \( R_{Path_{set}} \), is obtained. Next, we compute \( R_{Path_{set}} \), \( R_{Path_{set}}^{uc[k]} \) is obtained. Which is an iterative function reducing the input set to the \( k \) paths with highest sum of satisfaction values for \( u_c \). If the user has specified any other criteria, say \( u_c[j] \), then, in the subsequent iterations of the algorithm, we filter out paths from \( R_{Path_{set}}^{uc[k]} \) whose nodes do not satisfy the additional criteria (i.e., \( sat_{uc[j]} = 0 \)). Paths are filtered taking into account a single criterion per
each iteration, in the order of priority. The final set of ranked service combinations is sent as recommendations to the user.

Example 6: Assume the user assigns to $uc_1$, of Figure 1 highest priority. Let $k$ be set to 3. First, we compute the sum of satisfaction values of $uc_1$ for all four paths and we select the first 3 paths with maximum sum of satisfaction values and sort them, which are as follows: $R_{Path^{uc_1}}=\{P_2, P_3, P_4\}$. The input to the next iteration for finding the set of paths which satisfy the criteria $uc_2$ is $R_{Path^{uc_1}}$. The final set of paths are ranked according to the sum of satisfaction values of $uc_1$. The iterations for the remaining criteria ensure to select the paths whose services’ satisfaction value is not equal to 0. Hence, for $uc_3$, we verify that the nodes of paths in $R_{Path^{uc_1}}$ satisfy $uc_2$. The final ranking of the paths is $\{P_3, P_2, P_4\}$.

3) Global Evaluation: This strategy accounts for global user criteria, each denoted as $\pi$. For instance, a traveler may request to execute a service which execution fee is no higher than $50, or that it does not take more than 20 minutes of service processing time.

The strategy consists of integrating additional controls to the ranked paths obtained by executing either one of the above two strategies. The key steps to be executed are sketched as follows ($Set_P$ is a non empty set of paths in the composition graph $G$).

1) Compute $Set_P^\prime$ using Weighted Average of Criteria or Hierarchization of Objective Functions algorithm. Order $Set_P^\prime$ according to the paths’ PEF value.

2) Starting from the first path $P$ in $Set_P^\prime$ (i.e. highest ranked path), check for global thresholds. For instance, let $\varpi_P$ be the threshold value imposed on a property $c$. Let us assume, for simplicity, that the property value is expressed through an ordinal number, and that each Web service in the composition has an associated value $v$ for $c$. The satisfaction of $sat_{WS}(\varpi_P) = \sum_{j=1}^{\mid Set_P \mid} v_j \leq \varpi_P$ is checked, where $v_j$ is the property value for the service $j$ in the path $P$, and $\mid P.nodex \mid$ is the number of nodes in $P$.

3) If $sat_{WS}(\varpi_P) \neq 1$ compute $Set_P^\prime = Set_P^\prime \setminus P$

4) If more than one $\varpi_P$ is specified, repeat step (2), until no more $\varpi_P$ have to be evaluated.

5) Repeat from step (2) for the following $P \in Set_P^\prime$

We rank the paths obtained from the above strategy according to the PEF values obtained from either of the Weighted Average of Criteria or Hierarchization of Objective Functions strategy, which ever is used. Notice that step (2) can be replaced with other types of global controls, from a request to maintain a certain minimal satisfiability level, to a request of maximizing or minimizing a given criteria only. For example, a user may ask to obtain a high level of security, in addition to specifying detailed property conditions.

Example 7: Consider again the example from Figure 1, and additionally consider the global constraint the user specifies that $uc_1$, the total response time of all services in the composition to be less than 5 ms. The paths determined according to the weighted average method are: $\{P_3, P_2, P_1, P_4\}$. Let $P_3$, $P_2$, and $P_4$ satisfy the global constraint of the user. Hence, the set of ranked paths according to the local and global criteria of the user are $\{P_3, P_2, P_4\}$.

C. Ranking in case of equally satisfied criteria

If the user’s criteria are met equally by more than one possible (highly-ranked) compositions, the strategies will not produce any useful recommendation, in that the PEF value obtained by the strategies will result identical. This is also the case if the user entered criteria are fully met by the candidate service. To assist the user in identifying the best recommendations even in case of ties, WS-Rec further processes any user search criteria with identical satisfaction level across multiple combinations. Precisely, if the top ranked paths of a WS-Graph return the same PEF values, the user criteria with identical $Sat_{WS}(\cdot)$ value will be substituted to maximization and minimization functions, and a new PEF recomputed. This will result in a new order of paths, returning the best suitable path at the top of the ranking.

Substitution occurs for any Cond s.t. $Sat_{WS}(Cond) = 1$, the condition is mapped into a maximization or minimization operation. That is, if $Cond = \text{Att \ op \ V}$ and $\text{op} \geq \cdot$, then $Cond$ is mapped into $\text{Max(Att)}$ (or minimization, otherwise). Similarly, if $\text{op} = \leq \cdot$, $\text{op}$ is translated into a set maximization operator. Notice that converting into maximization or minimization criteria allows to measure the distance from the maximum value for the attribute domain (or minimum), therefore changing the value of $Sat_{WS}$, as it is computed according to a different rule (see Table 1).

Example 8: Let the PEF values of two paths $P_1$ and $P_2$ be 0.7. We find the set of capabilities that fully meet the user criteria. Let $ExecTime$ be the user criteria that any two nodes from $P_1$ and $P_2$ have fully met respectively. Let the user criteria be $ExecTime < 20s$, and let the capabilities of two nodes $N_1$ and $N_2$ in two different paths be $ExecTime = 10s$ and $ExecTime = 5s$. The satisfaction levels of the two nodes for $ExecTime$ are $sat_{N_1}(ExecTime) = 1$ and $sat_{N_2}(ExecTime) = 1$ respectively, since they fully meet the user criteria. We now map the user criteria into a minimization operation, $Min(ExecTime)$. We compute $Min(ExecTime)$ on the two nodes from $P_1$ and $P_2$. $Min(ExecTime)_{N_1} = 1 - \frac{11 - 10}{1000} = 0.991$ and $Min(ExecTime)_{N_2} = 1 - \frac{11 - 5}{1000} = 0.989$, where $MinDom(ExecTime) = 1$. Since $N_2$ has a better $ExecTime$ than $N_1$, that is, $Min(ExecTime)_{N_2} > Min(ExecTime)_{N_1}$, the path having $N_2$ is chosen over the path having $N_1$.

VI. EXPERIMENTAL RESULTS

Tests and implementation have been conducted on an Intel core 2 quad CPU, Q6700 @ 2.66GHz, 4 CPU cores, 3 GB RAM, Red Hat Enterprise Linux 5.6 operating system. In what follows, we report our results concerning the algorithms implemented by the Web service WS-Rec. We assessed both accuracy and overhead introduced by our ranking algorithms. For both types of tests, we compare our results with state-of-the-art algorithms using real world data sets.
We compared our recommendation algorithm with the Web service classification of the QWS dataset [1]. The dataset is a collection of quality of service information for 9 criteria of 365 Web services which are collected using a Web service Crawler Engine (WSCE) [2]. The classification is made according to the Web service Relevancy Function [1], which normalizes the service property values. The services are classified into four different classes: \{1,2,3,4\}, where 1 represents services of high quality, and 4 represents services of low quality. Given the advanced features of our system, we could compare the provided classification with the results obtained by running a simplified version of our first strategy, that is, the Weighted Average of Criteria. Since the QWS dataset does not account for composite Web services, we artificially created 20 service combinations from the dataset, taking into account functional requirements, when possible. Each combination consists of 4 services, combined together according to the inputs and outputs of the services, as a potential composite service. We use 9 different user criteria for ranking purposes. The aim is to maximize the satisfaction of the user’s criteria. To ensure fairness, we set the priority levels of all the criteria of the user as medium, as these are not included in the QWS classification. We tracked the average class of the four services of each combination from the ranked set of combinations (that is, each ranked combination is assigned an average class, based on the individual services’ original class). Our results show that the top 10 combinations ranked by our algorithm consistently have an average class value higher than those of the combinations ranked among the ranks 10 to 20, in line with the classification of the QWS dataset. For the second experiment, the input to the WS-Rec consists of 5 compositions of varying length (i.e., varying number of service providers). For example, 5 compositions of length 2 are: \{(WS_1, WS_2), (WS_3, WS_5), (WS_4, WS_6), (WS_7, WS_8), (WS_9, WS_10)\}. Each of the 5 compositions included services from the same class (per the classification provided from the QWS dataset). For example, we compute the average PEF of the combinations, \{(WS_1, WS_2), (WS_3, WS_5), (WS_4, WS_6), (WS_7, WS_8), (WS_9, WS_10)\} which belong to class 1, and similarly we consider combinations belonging to other classes and compute their average PEF. For each of the classes, we further varied the size of the composition graph, ranging from 10 to 100. As reported in Figure 2 (a), the set of combinations belonging to class 1, maintain higher average PEF values than the combinations belonging to other classes, regardless of the number of combinations. Hence, the results confirm that the recommender system ranks the services using a comparable order of the one used in the QWS dataset, and that the PEF is consistent regardless of the size of the combinations.

Performance Test. We tested the performance of our three strategies by running three separate tests each. Each test measures the time taken to rank and recommend service combinations to the user by varying the input size in terms of number of paths in the composition graph and number of user criteria. Each composition path consists of 4 services. As shown in Figure 2 (b), the time for ranking and recommending the Web services according to the user criteria, increases linearly as we increase the number of combinations input to WS-Rec. Also, the time increases linearly as we increase the number of user criteria. The overall results show that WS-Rec is extremely efficient: the time taken to compute the ranks for 90 paths is utmost 1 millisecond. That is, a graph with 360 nodes (i.e., Web services) is processed in only 1 ms. The results obtained for running the bounded values strategy are similar to the first strategy. The bounded value strategy in fact requires completing additional checks for satisfaction of global criteria, with a negligible additional time.

We compared the computation time of our ranking strategies with the integer programming (IP) method when it is used for ranking service compositions. Integer Programming is a popular method used to optimize a specific objective function which is subject to a set of linear constraints over integer variables. In the context of Web services, the integer programming method is used for selection of a service that best satisfies the given requirements or that has the best QoS attribute values. In order to compare both methods for generating ranked compositions, we generated all possible service combinations for a given user’s input and output. For instance, the user’s input and output might be \textit{location} and \textit{fee} respectively for a supply service, and all possible service combinations are generated for this input and output. These set of combinations are generated by syntactic matching of the inputs and outputs of the services. The combinations are provided to our ranking strategies as input. The output is a set of ranked combinations of services. In order to rank a set of service combinations using IP method, we ran the IP method for all possible candidate services for each task, as many times as the number of ranked service combinations desired in the output. For an input of 50 candidate services, and 4 services in each combination, that is, for a set of around \(6 \times 10^6\) service combinations, the IP method consumes time in the order of \(6 \times 10^6\) seconds, while our method consumes around 70 seconds. This result is obtained as the IP method generates only one service combination every time it is run. In order to rank \(n\) combinations, the IP method should be run \(n\) times, each time removing the combination resulted in previous run, which contributes to the high computation time for ranking. Hence, while the IP method is very efficient in generating a single service combination [20], it is not suitable for ranking service combinations, especially, when ranking of huge number of combinations is required.
VII. CONCLUSION

In this paper we presented a flexible framework for customized Web service recommendation. Our solution helps users determine best available services according to their needs, and takes into account both search criteria and service provisioning rules, in the case of both single and composite services. We presented an integrated architecture, and our results of our extensive experimental evaluation. We will extend the current solution along two dimensions. First, we will develop a framework that suggests security settings, by analyzing the candidate services, to the user which he or she will take into account in his or her security policies. Second, we will deploy user interfaces to facilitate user input and visualization of recommended services.

REFERENCES

APPENDIX

We discuss the privacy and security criteria in this appendix.

Privacy According to a general privacy policy model, relevant actors are the customers (WS users), the collectors (service providers) and possible third parties (either other service providers or a more general public) to which collectors may pass customers’ personal data. A customer may express a privacy preference stating what portion of his ‘information’ can be given out, to which other parties, for what purposes, and for how long the data retention is allowed. The user privacy preferences can then be matched against the collector’s privacy policies. Here, we assume the presence of a set of capabilities modeling relevant information of both privacy preference and policy. As an example, the AllowedInfo capability’s domain is ‘contact information’, ‘demographic information’, ‘financial information’, ‘medical information’, and ‘internet information’ (e.g., IP address, passwords, and cookies). Other relevant capabilities are the retention time (RetTime), allowed purpose (APurpose), and allowed third parties (AThirdParty). By using these capabilities and criteria conditions, users are able to express their privacy preference in terms of criteria conditions. These criteria are then matched against WS capabilities, so as to compute their levels of satisfaction.

Example 9: Assume that a user expresses the privacy preference: “contact and demographic information has to be retained only for 30 days, and no third parties releasing is allowed”. This preference can be modeled as $AllowedInfo_{\text{inf}}(\text{contactInfo}, \text{demographicInfo}) \land \text{RetTime}=30 \land A\text{Purpose}_{e}\{\text{none}\}$. Let us now assume that Web service $WS_1$ has capabilities: $AllowedInfo_{\text{inf}}(\text{contactInfo}, \text{demographicInfo}, \text{financialInfo}) \land \text{RetTime}=40 \land A\text{Purpose}_{e}\{\text{WS2}\}$. The corresponding levels of satisfaction are: $sat_{WS_1}(AllowedInfo_{\text{inf}}(\text{contactInfo}, \text{demographicInfo}))=0$; $sat_{WS_1}(\text{RetTime} =30) = 1 - \frac{40 - 30}{365} = 0.973$, where $\text{Dom}(\text{RetTime})=365$ days; $sat_{WS_1}(A\text{Purpose}_{e}\{\text{none}\})=0$. As such, $sat_{WS_1}(uc) = 0$.

Security Criteria To ensure secure and user-desirable compositions, our ranking system directly accounts for several security-related criteria. Specifically, the user can express more or less detailed security centered conditions. The taxonomy to specify user conditions, in this case, includes a set of known vulnerabilities of most common security technologies, as well as conventional security mechanisms (i.e., type of encryption supported and security standard mechanisms). The satisfaction level is then computed according to the mitigation techniques which are in use by the provider, and therefore whether certain security mechanisms are in place.

From an architectural standpoint, Web service security capabilities are expressed through SAML (Security Assertion Markup Language) assertions and matched for assessing their satisfaction level with respect to the user’s preferences. The SAML architecture relies on the presence of trusted authorities, issuing signed assertions on subjects (e.g., users, services, organizations), that is, a set of statements about the subject. In our approach, we assume the existence of a Secure Capability Authority (SCA) in charge of evaluating Web service security capabilities, and of issuing signed SAML assertions certifying such capabilities. In particular, we use the attribute statement of SAML assertions to express security capabilities of a Web service. According to the SAML specification, the attribute statement consists of an attribute name and an attribute value. We use the attribute name to denote the security vulnerability, whereas the attribute value gives information on how the security feature is mitigated by the corresponding Web service.

We consider the set of known security vulnerabilities or attacks and the corresponding known mitigation techniques (shown in the Table 1) to define the capabilities. These mitigation techniques are to be checked by the SCA to assess whether a vulnerability is addressed or not.

Similar to the case of privacy requirements, the consumer expresses user-criteria conditions toward preventing a certain attack or to check the support of certain security mechanisms. For example, consider a consumer who needs secure communications and wants the Web service to use the latest version of Secure Socket Layer (SSL). The satisfaction level measures the degree to which the security mechanisms of the service satisfy this condition. Intuitively, the satisfaction level would be 1 if the SSL is the latest version, and 0.8 if it is the last but one recent version. Similarly, if SSL is not supported at all, the satisfaction level would be set to 0.

Finally, if the user is unable to estimate the security guarantees offered by a composite Web service using fine-grained criteria, he may simply ask to maximize the composition’s security. The user can use the capability $Max(\text{Security})$, which implies the request to maximize all the security capabilities in the Security taxonomy. In all of the above cases, the system matches the security requirements of the consumer with the security capabilities of the service by examining the SAML assertions issued by the SCA.
<table>
<thead>
<tr>
<th>Vulnerability Type</th>
<th>Mitigation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CDATA Field Attack</strong></td>
<td>Validate the XML file using an XML schema</td>
</tr>
<tr>
<td>XML Injection Attack</td>
<td>Use Web service gateways</td>
</tr>
<tr>
<td>Cross Site Scripting</td>
<td></td>
</tr>
<tr>
<td>SQL Injection</td>
<td></td>
</tr>
<tr>
<td>XPath Injection</td>
<td></td>
</tr>
<tr>
<td><strong>XML Denial of Service</strong></td>
<td>When processing, use: Filters, XML gateways, XML parser options to prevent</td>
</tr>
<tr>
<td></td>
<td>them from processing malicious messages</td>
</tr>
<tr>
<td>Crafting XML messages with large</td>
<td></td>
</tr>
<tr>
<td>payloads, recursive content,</td>
<td></td>
</tr>
<tr>
<td>excessive nesting, malicious</td>
<td></td>
</tr>
<tr>
<td>external entities, malicious</td>
<td></td>
</tr>
<tr>
<td>DTDs</td>
<td></td>
</tr>
<tr>
<td><strong>Insecure Communications</strong></td>
<td>Use latest version of SSL or TLS</td>
</tr>
<tr>
<td></td>
<td>Use end to end security mechanisms like XML-encryption, XML-</td>
</tr>
<tr>
<td></td>
<td>signature, and SAML assertions</td>
</tr>
<tr>
<td><strong>Attacks on Integrity</strong></td>
<td>Have Web Service designs and configurations reviewed by a third party and</td>
</tr>
<tr>
<td></td>
<td>practice secure software development techniques</td>
</tr>
<tr>
<td>Parameter Tampering</td>
<td>Verification mechanism such as constraints on type and format in WSDL file</td>
</tr>
<tr>
<td>Schema Poisoning</td>
<td>Configure perimeter security correctly</td>
</tr>
<tr>
<td>Spoofing of UDDI Messages</td>
<td></td>
</tr>
<tr>
<td>Checksum Spoofing</td>
<td></td>
</tr>
<tr>
<td>Principal Spoofing</td>
<td></td>
</tr>
<tr>
<td>Routing Detours</td>
<td></td>
</tr>
<tr>
<td><strong>Attacks on Confidentiality</strong></td>
<td>Have Web Service designs and configurations reviewed by a third party and</td>
</tr>
<tr>
<td></td>
<td>practice secure software development techniques</td>
</tr>
<tr>
<td></td>
<td>Use SSL, HTTPS</td>
</tr>
<tr>
<td>Sniffing</td>
<td></td>
</tr>
<tr>
<td><strong>Privilege Escalation Attacks</strong></td>
<td>Difficult passwords, password strengthening rules</td>
</tr>
<tr>
<td>Dictionary Attack</td>
<td></td>
</tr>
<tr>
<td>Format String Attacks</td>
<td></td>
</tr>
<tr>
<td>Buffer Overflow Exploits</td>
<td></td>
</tr>
<tr>
<td>Race Conditions</td>
<td></td>
</tr>
</tbody>
</table>