Multicriteria Evaluation-Based Conceptual Framework for Composite Web Service Selection

Salem Chakhar¹, Serge Haddad², Lynda Mokdad¹, Vincent Mousseau³, and Samir Youcef³

¹ LAMSADE, Université Paris-Dauphine
Place du Maréchal de Lattre de Tassigny, 75775 Paris Cedex 16, France
{salem.chakhar,mokdad,samir.youcef}@lamsade.dauphine.fr
² Laboratoire Spécification et Vérification, École Normale Supérieure de Cachan
61, Avenue du Président Wilson 94235 Cachan Cedex, France
haddad@lsv.enscachan.fr
³ Laboratoire Génie Industriel, École Centrale Paris
Grande Voie des Vignes, 92295 Châtenay-Malabry Cedex, France
vincent.mousseau@ecp.fr

Abstract. The paper proposes a general framework to composite Web services selection based on multicriteria evaluation. The proposed framework extends the conventional Web services architecture by adding, in the registry, a new Multicriteria Evaluation Component (MEC) devoted to multicriteria evaluation. This additional component takes as input a set of composite Web services and a set of evaluation criteria and generates a set of recommended composite Web services. In addition to the description of the conceptual architecture of the framework, the paper also proposes solutions to construct and evaluate composite web services.

Keywords: Web service, Quality of Service, Multicriteria evaluation, Web service composition, Web service selection, Performance evaluation

1 Introduction

Individual web services are conceptually limited to relatively simple functionalities modeled through a collection of simple operations. However, for certain types of applications, it is necessary to combine a set of individual Web services to obtain more complex ones, called composite or aggregated Web services. One important issue within Web service composition is related to the selection of the most appropriate one among the different possible compositions. One possible solution is to use quality of service (QoS) to evaluate, compare and select the most appropriate composition(s). The QoS is defined as a combination of the different attributes of the Web services such as availability, response time, throughput, etc. The QoS is an important element of Web services and other modern technologies. Currently, most of 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 evaluation criterion or, at best, on a weighted sum of several quantitative evaluation criteria. Both evaluation schemas are not appropriate in practice since: (i) a single criterion does not permit to encompass all the facets of the problem, (ii) weighted sum-like aggregation rules may lead to the compensation problem since worst evaluations can be compensated by higher evaluations, and (iii) several QoS evaluation criteria are naturally qualitative ones but weighted sum-like aggregation rules cannot deal with this type of evaluation criteria.

The goal of this research is to propose a general framework to composite Web services selection based on multicriteria evaluation. The proposed framework extends the conventional Web services architecture by adding, in the registry, a new Multicriteria Evaluation Component (MEC) devoted to multicriteria evaluation. This additional component takes as input a set of composite Web services and a set of evaluation criteria. The output is a set of recommended composite Web services. The paper also proposes a solution to generate the different potential compositions which will be the main input for the MEC. Further, the paper shows how composite web services can be evaluated.

Two types of compositions are generally distinguished: static and dynamic. Static composition supposes that all Web services are predefined and that the composition graph (see Section 7) cannot change during execution. With dynamic composition, in turn, Web services are not predefined and composition graph may evolve over time. The dynamic composition is an important research issue. However, in this paper we assume a static composition environment. The extension to dynamic composition is under investigation.

The paper is organized as follows. Section 2 presents some related work. Section 3 introduces the notion of quality of service (QoS). Section 4 details the architecture of the proposed framework. Sections 5 and 6 further detail MEC and W-IRIS. The latter is a special kind of Web service used by MEC to infer the preference parameters needed to apply the multicriteria classification method—which is ELECTRE TRI in this paper. Section 7 shows how the set of potential composite Web services is constructed. Section 8 discusses the problem of composite Web service evaluation. Section 9 gives an illustrative application. Section 10 discusses some computational issues. Section 11 concludes the paper.

2 Related work

As underlined in the introduction, to choose among the different possible compositions, most of previous works use either a single QoS evaluation criterion or a weighted-sum of several quantitative QoS evaluation criteria. The following are some examples. The author in [14] considers two evaluation criteria (time and cost) and assigns to each one a weight between 0 and 1. The single combined score is computed as a weighted average of the scores of all attributes. The best composition of Web services can then be decided on the basis of the optimum
combined score. One important limitation of this proposal is the compensation problem mentioned earlier.

In [7], the service definition models the concept of “placeholder activity” to cater for dynamic composition of Web services. A placeholder activity is an abstract activity replaced on the fly with an effective activity. The author in [3] deals with dynamic service selection based on user requirement expressed in terms of a query language. In [8], the author considers the problem of dynamically selecting several alternative tasks within workflow using QoS evaluation. In [1], the service selection is performed locally based on a selection policy involving the parameters of the request, the characteristic of the services, the history of past executions and the status of the ongoing executions. One important shortcoming of [7][3][8][1] is the use of local selection strategy. In other terms, services are considered as independent. Within this strategy, there is no guarantee that the selected Web service is the best one.

To avoid the problem of sequential selection, Zeng et al. [23] propose the use of linear programming techniques to compute the “optimal” execution plans for composite Web service. However, the multi-attribute decision making approach used by the authors has the same limitation as weighted-sum aggregation rules, i.e., the compensation problem.

Maximilien and Singh [13] propose an ontology-based framework for dynamic Web service selection. However, they consider only a single criterion, which is not enough to take into account all the facets of the problem.

Menascé and Dubey [15] extends the work of Menascé et al. [16] on QoS brokering for service-oriented architectures (SOA) by designing, implementing, and experimentally evaluating a service selection QoS broker that maximizes a utility function for service consumers. These functions allow stakeholders to ascribe a value to the usefulness of a system as a function of several QoS criteria such as response time, throughput, and availability. This framework is very demanding in terms of preference information from the consumers. Indeed, consumer should provide to a QoS broker their utility functions and their cost constraints on the requested services. However, the most limitation of this work is the use of weighted-sum like optimization criterion, leading to compensation problem as mentioned earlier. One important finding of this paper is the use, by the QoS broker, of analytic queuing models to predict the QoS values of the various services that could be selected under varying workload conditions.

More recently, [10] use genetic algorithm for Web service selection with global QoS constraints. The authors integrate two policies (an enhanced initial policy and an evolution policy), which permits to overcome several shortcomings of genetic algorithm. The simulation on Web service selection shown an improved convergence and stability of genetic algorithm.

3 QoS evaluation criteria

Web services attributes can be organized into two families: functional requirements and non-functional requirements (NFRs). To evaluate the NFRs of a sys-
tems, several works have used the notion of QoS. Different definitions to the QoS have been proposed. Schmidt et al. [20] define QoS as the set of “all properties other than the functional behavior of an application”. In the most general case, this term refers to a set of requirements wished (or imposed) by a user (human being or software component) to the performance of an application during its execution [12].

There are several QoS evaluation criteria. A comprehensive list of commonly used criteria is given in Table 1. For each criterion, we provide a brief description, the type (quantitative or qualitative), and the preference direction where max means “the higher, the better” and min means “the lower, the better”.

As mentioned by [14], the exact definition and measurement process for each criterion must be well-defined to give service consumers and providers a common understanding. For ordinal QoS evaluation criteria, an ordinal measurement scale should be defined. A commonly used ordinal measurement scale is the Likert-type [21] one, which contains approximately equal number of favorable and unfavorable levels. An example is the five-points scale: very low, low, average, high, very high. These levels express difference in degrees but not quantities. The authorized mathematical operations on an ordinal scale are: “equal to” (=), “less than” (<), and “more than” (>).

Interval or ratio measurement scales need to be defined for quantitative QoS evaluation criteria. There are not generally accepted formula to measure Web services on quantitative QoS evaluation criteria. Response time, for example, could be measured as an average over the past 15 minutes, as a 90th percentile, or as an array of average times for each 15-minute interval during the day [14]. Klingemann et al. [9] use a continuous-time Markov chain to estimate the response time and the cost of a workflow.

In [2], the author discusses four QoS evaluation criteria (response-time, cost, reliability and fidelity) and proposes different ways to use them to evaluate composite web services.

Maximilien and Singh [13] proposed an ontology-based framework for dynamic Web service selection that provides a starting point for a QoS lingua franca. However, they did not address the fact that some QoS metrics, such as response time, depend on workload intensity level, which means a single value is not appropriate [14].

Yousef et al. [22] propose a simulation-based response-time evaluation.

4 Extended Web service architecture

We first present the conventional Web service architecture. Then, we introduce the proposed architecture. Next, we detail the different XML schema for information exchange among the entities involved in the extended Web service model.
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Preference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>The degree to which a system, subsystem, or equipment is operable and in a committable state at the start of a mission, when the mission is called for at an unknown, i.e., a random, time. In others terms, availability is the proportion of time a system is functional.</td>
<td>Quantitative</td>
<td>max</td>
</tr>
<tr>
<td>Response time</td>
<td>The lap of time from request sending to response reception.</td>
<td>Quantitative</td>
<td>min</td>
</tr>
<tr>
<td>Throughput</td>
<td>The rate at which a service can process requests.</td>
<td>Quantitative</td>
<td>max</td>
</tr>
<tr>
<td>Reliability</td>
<td>The likelihood of success using a service.</td>
<td>Quantitative</td>
<td>max</td>
</tr>
<tr>
<td>Security</td>
<td>It captures the level and kind of security a service provides.</td>
<td>Qualitative</td>
<td>max</td>
</tr>
<tr>
<td>Robustness</td>
<td>The degree to which a system or component can function correctly in the presence of invalid inputs or stressful environment conditions.</td>
<td>Qualitative</td>
<td>max</td>
</tr>
<tr>
<td>Scalability</td>
<td>It defines whether the service capacities can be increased as needed.</td>
<td>Qualitative</td>
<td>max</td>
</tr>
<tr>
<td>Integrity</td>
<td>The quality aspect of how the Web service maintains the correctness of the interaction in respect to the source. Proper execution of Web service transactions will provide the correctness of interaction. A transaction refers to a sequence of activities to be treated as a single unit of work. All the activities have to be completed to make the transaction successful. When a transaction does not complete, all the changes made are rolled back.</td>
<td>Qualitative</td>
<td>max</td>
</tr>
<tr>
<td>Reputation</td>
<td>It is a measure of trustworthiness. It mainly depends on end user’s experiences of using a service.</td>
<td>Qualitative</td>
<td>max</td>
</tr>
<tr>
<td>Latency</td>
<td>The amount of time it takes a packet to travel from web service to another web service.</td>
<td>Quantitative</td>
<td>min</td>
</tr>
<tr>
<td>Accuracy</td>
<td>Represents the error rate generated by the Web service. It can be measured by the numbers of errors generated in a certain time interval.</td>
<td>Quantitative</td>
<td>max</td>
</tr>
<tr>
<td>Regulatory</td>
<td>The quality aspect of the Web service according to rules, law, compliance with standards, and the established service level agreement. Strict adherence to correct versions of standards by service providers is necessary for proper invocation of Web services by service requestors.</td>
<td>Qualitative</td>
<td>max</td>
</tr>
<tr>
<td>Authentication</td>
<td>The capacity of a service to authenticate other entities—users or other Web services—in order to access them.</td>
<td>Qualitative</td>
<td>max</td>
</tr>
<tr>
<td>Confidentiality</td>
<td>The capacity that a Web service respect that a given data should be treated properly, so that only authorized entities (or Web services) can access or modify the data.</td>
<td>Qualitative</td>
<td>max</td>
</tr>
<tr>
<td>Traceability</td>
<td>The capacity that a web service traces itself history when a request was serviced.</td>
<td>Qualitative</td>
<td>max</td>
</tr>
<tr>
<td>Auditability</td>
<td>The capacity that a Web service encrypts data.</td>
<td>Qualitative</td>
<td>max</td>
</tr>
<tr>
<td>Non-repudiation</td>
<td>The fact that an entity (service) cannot deny requesting a service after the fact.</td>
<td>Qualitative</td>
<td>max</td>
</tr>
<tr>
<td>Accessibility</td>
<td>The degree that a Web service is capable of serving a Web service request. It may be expressed as a probability measure denoting the success rate or chance of a successful service instantiation at a point in time. There could be situations when a Web service is available but not accessible.</td>
<td>Qualitative</td>
<td>max</td>
</tr>
<tr>
<td>Cost</td>
<td>Web service cost specification.</td>
<td>Quantitative</td>
<td>min</td>
</tr>
</tbody>
</table>

Table 1. List of QoS evaluation criteria
4.1 Conventional Web service architecture

The Web service architecture is defined by 3WC in order to determinate a common set of concepts and relationships that allow different implementations working together [4]. Figure 1 shows a graphical representation of the traditional Web service architecture. The conventional Web service architecture consists of three entities, the service provider, the service registry and the service consumer. The service provider creates or simply offers the Web service. The service provider needs to describe the Web service in a standard format, which is often XML, and publish it in a central service registry. The service registry contains additional information about the service provider, such as address and contact of the providing company, and technical details about the service. The service consumer retrieves the information from the registry and uses the service description obtained to bind to and invoke the Web service. The appropriate methods are depicted in Figure 1 by the keywords “publish”, “bind” and “find”.

Web services architecture is loosely coupled, service oriented. The Web Service Description Language (WSDL) uses the XML format to describe the methods provided by a Web service, including input and output parameters, data types and the transport protocol, which is typically HTTP, to be used. The Universal Description Discovery and Integration standard (UDDI) suggests means to publish details about a service provider, the services that are stored and the opportunity for service consumers to find service providers and Web service details. The Simple Object Access Protocol (SOAP) is generally used for XML formatted information exchange among the entities involved in the Web service model.

4.2 Proposed Web service architecture

The proposed framework extends the conventional Web services architecture by adding, in the registry, a new Multicriteria Evaluation Component (MEC) devoted to multicriteria evaluation. The general schema of the extended architecture is given in Figure 2. According to the requirement of the consumer, the registry opts either for conventional evaluation or for multicriteria evaluation.
By default, the registry uses conventional evaluation; multicriteria evaluation is used only if the consumer explicitly specifies this in the SOAP message addressed to the registry. This ensures the flexibility of the proposed architecture.

The application of a multicriteria method needs the definition of a set of preference parameters. The definition of these parameters needs an important cognitive effort from the consumer. To reduce this effort, MEC uses specific Web service called W-IRIS which is a Web version of IRIS (Interactive Robustness analysis and Parameters Inference for multicriteria Sorting Problems) [5] system permitting to infer the different preference parameters.

As we can see in Figure 2, the three basic operations denoted by “publish”, “bind” and “find” still exist. Two additional operations, denoted by keywords “infer” and “evaluate” are included in the extended architecture. The first permits to handle data exchange between MEC and W-IRIS. The latter permits to handle data exchange between MEC and DecisionDeck platform, presented in the end of this subsection.

To achieve the interaction among the entities of the extended Web service model, we need to extend some SOAP protocols and add new ones. More specifically, we need to extend protocols of consumer request to registry and registry response to consumer; and add the ones relative to MEC request to W-IRIS and W-IRIS response to MEC. A detailed description of the proposed architecture is given in Figure 3.
Fig. 3. Dynamics of the system
One important remark to evoke at this level is relative to the fact that several providers may be implied by the consumer or by the registry and that a given provider may invoke other providers. This is illustrated by the discounted arrows between provider-1 and provider-N in Figure 3. It is important to mention that the proposed architecture still applies since the multicriteria evaluation implies only the providers directly invoked by the consumer or the registry. In addition, we suppose that the evaluations of the services proposed by directly invoked providers include the evaluations of the ones indirectly invoked by these providers.

As we will be explained latter, W-IRIS permits to infer the different preference parameters needed to apply multicriteria evaluation using ELECTRE TRI method. The inference procedure included in W-IRIS needs the resolution of different mathematical programs. For this purpose, W-IRIS includes the solver GLPK, which is an open-source and free package (see [11]).

The current version of MEC supports the advanced multicriteria method ELECTRE TRI (see [6]) and several elementary methods (weighted sum, conjunctive and disjunctive rules and the majority rule). Additional methods will be included in the future via the DecisionDeck platform. The DecisionDeck platform, which is under development, is issued from D2-Decision Deck project that has started in 2003 under the name EVAL, an acronym which refers to an ongoing research project funded by the Government of the Walloon region (Belgium). The aim is to develop a Web-based platform to assist decision makers in evaluating alternatives in a multicriteria and multi-experts context. The EVAL platform is currently available on the collaborative development Web site www.sourceforge.org.

In the rest of this section, we detail the required extension/addition to support data exchange between the different entities of the proposed architecture.

### 4.3 Consumer—Registry communication

The XML schema of consumer request to registry is given in Figure 4. The consumer may choose between two types of evaluations: mono-criterion or multicriteria. This ensured by the `<choice>` tag in Figure 4. The monocriterion-evaluation element in this figure refers to conventional evaluation. It will not be detailed here. The multicriteria-evaluation element corresponds to multicriteria evaluation. It is a complex type composed of four elements: result_type, evaluation_criteria, sorting_data, and parameters.

**Type of result** `<result_type> tag` The consumer may indicate the type of result of multicriteria evaluation, which may be “choice”, “ranging” or “sorting” (see Section 5.6). The default value is “choice” indicating that a restricted subset of compositions will be returned to the consumer.

**Evaluation criteria** `<evaluation_criteria> tag` The consumer must indicate at least two QoS evaluation criteria to be used to evaluate and compare the different potential compositions. For each criterion, it specifies (i) the name, which may
be any one from the list given in Table 1; and (ii) zero or several preference parameters. For each preference parameter, it indicates the name and the value. The name is one of the following list:

- Weight
- Indifference threshold
- Preference threshold
- Veto threshold
- Aspiration level
- Reservation level

<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">
  <xsd:complexType element name="find_service">
    <xsd:choice>
      <xsd:element name="monocriterion_evaluation" type="monoType"/>
      <xsd:element name="multicriteria_evaluation" type="mecType"/>
    </xsd:choice>
  </xsd:complexType>
  <xsd:complexType name="monoType">
    ...
  </xsd:complexType>
  <xsd:complexType name="mecType">
    <xsd:sequence>
      <xsd:element name="result_type" type="token" #REQUIRED/>
      <xsd:element name="evaluation_criteria">
        <xsd:element name="sorting_data">
          <xsd:element name="parameters">
            ...
          </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="evaluation_criteria">
          <xsd:sequence>
            <xsd:group ref="criteriaGroup">
              ...
            </xsd:sequence>
          </xsd:complexType>
          <xsd:group name="criteriaGroup">
            ...
          </xsd:group>
        </xsd:complexType>
        <xsd:complexType name="criterionType">
          ...
        </xsd:complexType>
        <xsd:complexType name="preference_parameters">
          ...
        </xsd:complexType>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:complexType>
  <xsd:complexType name="preference_parameters">
    ...
  </xsd:complexType>
</xsd:schema>

Fig. 4. XML schema of consumer request to registry (continued on the next page)
The weights represent the relative importance of the different QoS evaluation criterion according to the aspiration of the consumer. The next three parameters are often used within outranking relation-based multicriteria methods (see [6]).
The **indifference** and **preference thresholds** are used to model the imprecision and uncertainty in the consumer preferences. The **veto threshold** is often used to compute the **discordance index** as will be explained latter. The **aspiration level** is the minimal (maximal, resp.) value defined on an evaluation criterion and which should be exceeded (resp. not exceeded) by each composition to be acceptable. One or several aspiration levels may be defined for different criteria. The **reservation level** represents the minimal value on a given criterion that should be verified by any potential composition. The reservation level needs to be defined for all the evaluation criteria.

**Sorting data** (<sorting-data> tag) It applies when the type of result is “sorting”, i.e., a classification of different composite Web services into a set of predefined categories (see Section 5.6). These categories are defined in terms of a set of profile limits representing the boundaries between these categories. The consumer should provide: (i) the number of categories; (ii) the profile limits; and (iii) a set of assignment examples. The number of categories is optional. The default value for the number of categories is 3. The profile limits may be either provided by the consumer or generated automatically, as explained in Section 4.4. This second option is included to reduce the cognitive effort of the consumer.

A profile limit is defined as a vector of \( m \) elements, where \( m \) is the number of evaluation criteria. An advanced definition of a profile may need the use of indifference threshold, preference threshold and/or veto threshold.

The consumer should also provide a set of assignment examples which will be used to infer the preference parameters. More details on how these assignment examples are defined and used will be given in Section 4.4.

**Parameters** (<parameters> tag) This optional element is used to cope with some multicriteria methods that require the definition of some technical parameters. For instance the **cutting level** is used with some multicriteria outranking methods to validate or invalidate the outranking relation. Note that this parameter should belongs to \([0, 1]\) and when omitted, the value 0.5 is used.

The XML schema of registry response to consumer is given in Figure 5. The registry indicates the list of recommended compositions. As it is shown in Figure 5, three types of recommendations are possible: choice, ranging or classification. This is ensured by the <choice> tag in Figure 5. A brief description of these different types of recommendation follows.

- **List of best compositions.** This applies when the type of result is “choice”. In this case, the recommendation of MEC is a restricted set of equivalent compositions from which the user should choose one for application.

- **A ranging of compositions.** This applies when the type of result is “ranging”. The recommendation of MEC in this case is a list of compositions ordered from the best to the worst. These information are handled by the <order> element in Figure 5.

- **A classification of compositions.** This last case applies when the type of result is “sorting”. The output of MEC is a list of compositions, each one
is characterized by its category indicated by the <category> element in Figure 5.

Fig. 5. XML schema of registry response to consumer
4.4 MEC—W-IRIS communication

The W-IRIS is a special kind of Web service used by MEC to infer the preference parameters to use with ELECTRE TRI method. An brief overview of ELECTRE TRI is provided in Appendix A. This method is used by MEC to assign composite Web services into different categories. It applies when “type of result” in the SOAP message sent by the consumer to registry is “sorting” (see Figure 4). The XML schema of the “infer” SOAP message sent by the MEC to W-IRIS is given in Figure 6. It contains the same information included in the “sorting_data” element given in Figure 4. To avoid redundancy, the “sorting_data” is not detailed in Figure 6.

In the most general case, the inputs of W-IRIS are: (i) the number of categories, (ii) a set of profile limits, and (iii) a set of assignment examples. All of these data are extracted from the SOAP message sent by the consumer to registry detailed in the previous subsection (see Figure 4). As underlined earlier, the number of categories is an optional parameter and when it is omitted, three categories are automatically used. These categories are denoted $C_1$, $C_2$, and $C_3$. Category $C_3$ corresponds to recommended compositions and category $C_1$ corresponds to unrecommended compositions. Category $C_2$ corresponds to intermediate compositions. As introduced above, the categories are defined in terms of a set of profile limits. Figure 7 shows the definition of categories $C_1$, $C_2$, and $C_3$ in terms of two profile limits $b_1$ and $b_2$. Each profile is defined as a vector of $m$ elements where $m$ is the number of considered QoS evaluation criteria, denoted $g_1$, $g_2$, $\ldots$, $g_m$ in Figure 7. Profiles $b_0$ and $b_4$ are defined as a vector of lower and higher boundaries of evaluation criteria scales.

In the case where the profile limits are not provided by the consumer, they will be automatically constructed by MEC. To this purpose, the measurement scale of each QoS evaluation criterion included in the “find” SOAP message sent by the consumer to the registry is subdivided into three equal intervals. Then, profile limits are defined by joining the limits of these intervals on the different evaluation criteria.

Fig. 6. XML schema of MEC request to W-IRIS
The set of assignment examples are defined as follows. First, MEC generates a set of fictive compositions. Each fictive composition \( k_f \) is associated with a vector of \( m \) elements:

\[
(g_1(k_f), g_2(k_f), \cdots, g_m(k_f)),
\]

where \( m \) is the number of QoS evaluation criteria. Evaluations \( g_j(k_f) \) \( (j = 1, \cdots, m) \) are defined such that \( k_f \) may be assigned to two successive categories. For better explanation, consider two categories \( C_i \) and \( C_j \) and let \( b_h \) be the profile limit between \( C_i \) and \( C_j \) with evaluation vector \( (g_1(b_h), g_2(b_h), \cdots, g_m(b_h)) \). Then, a fictive composition \( k_f \) is defined such that its performances on a subset of QoS evaluation permits to assign it to \( C_i \) and the rest permits to assign it to \( C_j \).

![Diagram](image)

Fig. 7. Definition of classes \( C_1, C_2 \) and \( C_3 \) in terms of the profile limits

XML schema of W-IRIS “inference_output” SOAP message to MEC is given in Figures 8. It is a collection of preference parameters and the corresponding values. These parameters will be used by MEC to apply ELECTRE TRI.

5 Multicriteria evaluation component

The general schema of multicriteria evaluation component (MEC) is depicted in Figure 2. Basically, it takes as input a set of composite Web services and a set of QoS evaluation criteria and generates a set of recommended compositions. The final choice should be performed by the consumer, based on the MEC recommendation.

In the rest of the paper, \( K = \{k_1, k_2, \cdots, k_n\} \) denotes a set of \( n \) potential composite Web services and \( I = \{1, 2, \cdots, n\} \) denotes the indices of these services. The solution proposed to construct set \( K \) will be detailed in Section 7.
5.1 Definition of QoS evaluation criteria

The set of QoS evaluation criteria to be used is extracted from the “find” SOAP message sent by the consumer to the registry (see Section 4.3). At least two QoS evaluation criteria should be provided. The set of evaluation criteria will be denoted by \( F = \{g_1, g_2, \cdots, g_m\} \) in the rest of the paper. Theoretically, there is no limit to the number of criteria. We observe, however, that a large set increases the cognitive effort required from the consumer and a few ones do not permit to encompass all the facets of the selection problem.

5.2 Quantification of evaluation criteria

Quantification permits to transform qualitative evaluation criteria into quantitative ones by assigning values to the qualitative data. This is useful for mostly of multicriteria methods based on weighted-sum like aggregation decision rules. The most used quantification method is the scaling one. The quantification process consists in the definition of a measurement scale as the one mentioned earlier and then to associate to each level of the scale a numerical value. For example, the numbers 1, 2, 3, 4 and 5 may be associated to the fifth levels scale introduced in Section 3, going from very low to very high. Note, however, that the set of mathematical operations authorized is the same as the one mentioned in Section 3, i.e., “equal to” (=), “less than” (<), “more than” (>).

5.3 Generation of performance table

Once potential composite Web services are constructed and evaluation criteria are identified, the next step consists in the evaluation of all these composite Web services against all the evaluation criteria in \( F \). The evaluation of a composite Web Service \( k_i \in K \) in respect to criterion \( g_j \in F \) is denoted \( g_j(k_i) \). The matrix \( [g_j(k_i)], \forall i \in I, \forall j \in F \) is called the performance table. The general structure
of the performance table is given in Figure 9. The computing of $g_j(k_i), \forall i \in I, \forall j \in F$, will be dealt with in Section 8.

<table>
<thead>
<tr>
<th>Potential composite Web services</th>
<th>QoS evaluation criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_1$</td>
<td>$g_1(k_1)$ $\cdots$ $g_m(k_1)$</td>
</tr>
<tr>
<td>$\cdots$</td>
<td>$\cdots$ $\cdots$ $\cdots$</td>
</tr>
<tr>
<td>$k_n$</td>
<td>$g_1(k_n)$ $\cdots$ $g_m(k_n)$</td>
</tr>
</tbody>
</table>

Fig. 9. General structure of performance table

5.4 Definition of preference parameters

Most of multicriteria methods require the definition of a set of preference parameters. Two cases hold here: either the preference parameters are provided explicitly by the consumer and extracted from the “find” SOAP message to the registry; or inferred by W-IRIS based on the assignment examples equally extracted from the “find” SOAP message sent by the consumer.

5.5 Multicriteria evaluation

The input for this step are the performance table and the preference parameters. The objective of multicriteria evaluation is to evaluate and compare the different compositions in $K$.

As signaled above the advanced multicriteria method ELECTRE TRI and four elementary methods (weighted sum, conjunctive and disjunctive rules, and the majority rule) are incorporated in the framework. Additional methods will be included in the future. The application of ELECTRE TRI is reported in Section 9.

5.6 Recommendation

As underlined above, three types of recommendations are possible within the proposed framework. Based on the specifications of the consumer, one of the following results is provided to it:

- one or a restricted set of composite Web services;
- a ranging of composite Web services from best to worst with eventually equal positions;
- a classification of composite Web services into different pre-defined categories.

These three types of result correspond in fact to the three ways usually used to formalize multicriteria problems as identified by [19]: choice, ranking and sorting.
6 W-IRIS Web service

W-IRIS implements a Web version of IRIS (Interactive Robustness analysis and Parameters Inference for multicriteria Sorting Problems) system. IRIS supports a methodology of inference initially proposed in [18] (see also [17]). Here, we will shortly introduce the principle of the inference procedure included in W-IRIS.

The input for the inference procedure is the set of assignment examples provided by the consumer. Let \( K^* \) be the set of these assignment examples. We define two sets
\[
S^+ = \{(k, b_h) \in K^* \times B: \text{that the consumer states that } kSb_h\}
\]
and
\[
S^- = \{(k, b_h) \in K^* \times B: \text{that the consumer states that } \neg(kSb_h)\}.
\]
Relation \( S \) is the outranking binary relation (see Appendix A) defined such that \( kSb_h \) means that the evaluation of composition \( k \) upon all evaluation criteria is at least as good as the evaluation of the profile \( b_h \) (lower limit of category \( C_{h+1} \)). The idea of the inference procedure consists in searching a set of preference parameters that permit to re-do the assignment examples provided by the consumer. This can be obtained by resolving the following system:

\[
\begin{align*}
\sigma(k, b_h) &\geq \lambda, & \forall (k, b_h) \in S^+ \\
\sigma(k, b_h) &< \lambda, & \forall (k, b_h) \in S^- \\
\lambda &\in [0.5, 1] \\
q_j(b_h) &\leq p_j(b_h) \leq v_j(b_h), & \forall (j, h) \in F \times B \\
\sum_{j=1}^m w_j &= 1; w_j \geq 0, & \forall j \in F
\end{align*}
\]

where:

- \( \sigma(k, b_h) \in [0, 1] \) is the credibility degree that measures the extent to which composition \( k \) outranks profile limit \( b_h \);
- \( \lambda \) is the cutting level used to validate or invalidate the outranking relation;
- \( q_j(b_h), p_j(b_h) \) and \( v_j(b_h) \) (\( j = 1, \ldots, m \)) are the indifference, preference and veto threshold associated with profile limit \( b_h \); and
- \( w_j \) (\( j = 1, \ldots, m \)) is the weights of evaluation criteria.

Some of these concepts are introduced in Section 4.3. Further details are given in Appendix A, along with the presentation of ELECTRE TRI method. This system can be expressed through a mathematical program having as variables the parameters to infer. Then, the values for these parameters are obtained by maximizing the minium slack for this system of constraints. The inferred parameters are then used to apply ELECTRE TRI.

To construct the system, W-IRIS uses the data extracted from the XML document corresponding to the “evaluate” request sent by MEC to W-IRIS. Then W-IRIS uses the routines of GLPK\(^1\) to resolve the mathematical program issued from the system above.

\(^1\) More information on GLPK solver and routines is available at http://www.gnu.org/software/ glpk/.
7 Constructing potential composite Web services

Following Menascé [14], a Web service is defined as follows.

Definition 1 A Web service $S_i$ is a tuple $(F_i, Q_i, H_i)$, where:

- $F_i$ is a description of the service’s functionality,
- $Q_i$ is a specification of its QoS evaluation criteria, and
- $H_i$ is its cost specification.

Menascé [14] uses the term “attribute” instead of “criteria”. The last one is more general and hence is adopted here. A Web service’s QoS evaluation criterion may be any one of the list provided in Table 1. A Web service’s cost is often related to its quality. Faster, reliable, secure services will be more expensive, for example, but there could also be penalties associated with not meeting certain QoS goals or service-level agreements (SLAs) [14].

A composition operation implies several individual Web services. The relationships among the individual Web services may be represented by a connected and directed graph $G = (X, V)$ where $X = \{S_1, S_2, \ldots, S_m\}$ is the set of individual Web services and $V = \{(S_i, S_j) : S_i, S_j \in X \land S_i \ can \ invoke \ S_j\}$. $G = (X, V)$ is called the composition graph. Figure 10, which is reproduced from [14], presents a composition graph example implying six individual Web services $S_1, S_2, S_3, S_4, S_5$ and $S_6$.

![Composition Graph Example](image)

**Fig. 10.** An example of composition graph

The arcs in this figure represent different types of invocation. These last ones correspond to different BPEL constructors. The basic BPEL constructors are:

- **Sequential invocation.** A Web service is activated as a result of the completion of one of a set of mutually exclusive predecessor activities. These activities may be listed with the XML `<sequence>` tag, that is, in lexical order. Example: $S_2$ and $S_6$ in Figure 10.
• **Parallel invocation** (fork). It represents a point in the process where a single thread of control splits into multiple threads of control which can be executed in parallel. This pattern is supported by BPEL using XML `<flow>` tag. **Example:** $S_1$ in Figure 10 which can invoke $S_2$ and $S_3$ in parallel.

• **Probabilistic invocation.** A probability value $p$ on an outgoing arrow from $S_i$ to $S_j$ indicates that $S_i$ invokes $S_j$ with probability $p$. If no value is indicated, the probability is assumed to be 1. **Example:** $S_3$ in Figure 10 which can invoke $S_4$ and $S_5$ with probability $p_1$ or $S_5$ with probability $p_2$.

• **Conditional invocation.** Represents a situation where one or several branches are chosen. The first situation can directly be implemented using `<switch>` constructor and the second through control links inherited from XLANG. There is no example of conditional invocation in Figure 10.

• **Synchronized invocation (join).** A Web service is activated only when all of its predecessor Web services have completed. It can be implemented using control links inherited from XLANG. **Example:** $S_6$ in Figure 10 which requires the completion of Web services $S_4$ and $S_5$.

We assume that each Web service $S_i$ has a unique functionality $F_i$. In turn, the same functionality may be provided by different providers. Let $P_i$ be the collection of providers supporting functionality $F_i$ of Web service $S_i$: $P_i = \{s_{i1}^1, s_{i2}^1, \ldots, s_{in}^i\}$ where $n_i$ is the number of providers in $P_i$. A composite Web service is defined as follows.

**Definition 2** Let $S_1, S_2, \ldots, S_n$ be a set of $n$ individual Web services such that $S_i = (F_i, Q_i, H_i)$ $(i = 1, \ldots, n)$. Let $P_i$ be the collection of Web services supporting functionality $F_i$. Let $G = (X, V)$ be the composition graph associated with $S_1, S_2, \ldots, S_n$. A **composite Web service** $k$ is an instance $\{s_1, s_2, \ldots, s_n\}$ of $G$ defined such that $s_1 \in P_1, s_2 \in P_2, \ldots, s_n \in P_n$.

It is clear that this definition may lead to a large number of compositions. Some solutions to avoid this problem will be introduced in Section 10.

To take into account the invocation probabilities associated with some Web services, we define a new function, called $\pi$, as follows:

$$\pi: X \times X \rightarrow [0, 1]$$

$$S_i \times S_j \rightarrow \pi(S_i, S_j)$$

The number $\pi(S_i, S_j)$ represents the probability that $S_i$ invokes $S_j$.

**Example 1.** Consider the graph of Figure 10 and suppose that:

- $P_1 = \{s_1^1, s_2^1, s_3^1, s_4^1\}$
- $P_2 = \{s_1^2, s_2^2, s_3^2\}$
- $P_3 = \{s_1^3\}$
- $P_4 = \{s_1^4\}$
- $P_5 = \{s_1^5, s_2^5, s_3^5, s_4^5, s_5^5\}$
- $P_6 = \{s_1^6, s_2^6\}$
where \( s^j_i \) is the \( j \)th provider of Web service \( S_i \). Then, the following are some composite Web services:

- \( k_{49} \) associated with \( G_{49} = (X_{49}, V_{49}) \) where:
  - \( X_{49} = \{s^1_1, s^2_2, s^3_3, s^4_4, s^5_5, s^6_6\} \)
  - \( V_{49} = \{(s^1_1, s^2_2); (s^1_1, s^3_3); (s^2_2, s^4_4); (s^3_3, s^5_5); (s^4_4, s^6_6); (s^5_5, s^6_6)\} \)
  - \( \pi(s^1_1, s^2_2) = p_1; \pi(s^3_3, s^6_6) = p_2 \)

- \( k_{112} \) associated with \( G_{112} = (X_{112}, V_{112}) \) where:
  - \( X_{112} = \{s^1_1, s^2_2, s^3_3, s^4_4, s^5_5, s^6_6\} \)
  - \( V_{112} = \{(s^1_1, s^2_2); (s^1_1, s^3_3); (s^2_2, s^4_4); (s^3_3, s^5_5); (s^4_4, s^6_6); (s^5_5, s^6_6)\} \)
  - \( \pi(s^3_3, s^4_4) = p_1; \pi(s^2_2, s^5_5) = p_2 \)

- \( k_{185} \) associated with \( G_{185} = (X_{185}, V_{185}) \) where:
  - \( X_{185} = \{s^1_1, s^2_2, s^3_3, s^4_4, s^5_5, s^6_6\} \)
  - \( V_{185} = \{(s^1_1, s^2_2); (s^1_1, s^3_3); (s^2_2, s^4_4); (s^3_3, s^5_5); (s^4_4, s^6_6); (s^5_5, s^6_6)\} \)
  - \( \pi(s^3_3, s^4_4) = p_1; \pi(s^2_2, s^5_5) = p_2 \)

Figure 11 shows graphically the composite Web services \( k_{49}, k_{112} \) and \( k_{185} \).

![Figure 11. Composed Web services \( k_{49}, k_{112} \) and \( k_{185} \)](image)

To construct the set of potential compositions, we have incorporated two algorithms in the MEC. The first one, called CompositionGraph and given below,
permits the construction of the composition graph. In this algorithm, $\Gamma^+(x)$ returns the set of successors of node $x$: $\Gamma^+(x) = \{y \in X : (x, y) \in V\}$. The input of $\text{CompositionGraph}$ algorithm is the description of the different individual Web services stored in the UDDI registry and the invocation probability function $\pi$. The output is the composition graph $G$ defined earlier. The algorithm runs in $O(n^2)$ where $n$ is the number of Web services.

### Algorithm CompositionGraph

**INPUT:** $S = S_1, S_2, \ldots, S_m$: the set of Web services  
**OUTPUT:** $G = (X, V)$: composition graph  

$x \leftarrow S$  
$z \leftarrow S$  
curr_node $\leftarrow S_1$  

WHILE $Z \neq \emptyset$

FOR each $s_j \in X$

IF $s_j \in \Gamma^+(\text{curr} \_ \text{node})$

THEN $V \leftarrow (\text{curr} \_ \text{node}, s_j)$

END FOR

$Z \leftarrow Z \ \backslash \ \text{curr} \_ \text{node}$  
curr_node $\leftarrow$ pick a node in $Z$

END WHILE

The second algorithm, given hereafter, is $\text{CompositionsConstruction}$ that generates the potential compositions. The algorithm $\text{CompositionsConstruction}$ proceeds as follows. First a tree $T$ is constructed using $\text{Construct} \_ \text{Tree}$. The inputs for this procedure is the set of nodes $X$ and the set of providers for each node in $X$: $P = \{P_1, P_2, \ldots, P_n\}$. The tree $T$ is constructed as follows. The nodes of the $i$th level are the providers in $P_i$. For each node in level $i$, we associate the providers in set $P_{i+1}$ as sons. The same reasoning is used for $i = 1$ to $n - 1$. The nodes of the $n - 1$th level is associated with the providers in $P_n$. Finally, a root $r$ is added to $T$ as the parent of nodes in the first level (representing the providers in $P_1$). Then, the set of nodes for each composition is obtained as an elementary path in $T$. Figure 12 shows a schematic representation of the tree associated with composition graph given in Figure 10. The first elementary path is composed of $s_1^1, s_2^1, s_3^1, s_4^1$ and $s_0$. The last elementary path is $s_1^4, s_2^4, s_3^4, s_4^4, s_5^4$ and $s_6^4$. Thus, the node set of $k_1$ is $X_1 = \{s_1^1, s_2^1, s_3^1, s_4^1, s_5^1, s_0\}$ and the node set of $k_{240}$ is $X_{240} = \{s_1^4, s_2^4, s_3^4, s_4^4, s_5^4, s_6^4\}$.
Once the collections of providers of the different compositions are defined, algorithm CompositionsConstruction use the composition graph $G$ to construct the different compositions as instances of graph $G$.

The complexity of algorithm CompositionsConstruction is $O(r_1 \times (r_2 + r_3))$ where $r_1 = |V|$ is the cardinality of $V$, $r_2 = \prod_{i=1}^{n} |P_i|$ is the number of compositions and $r_3$ is the complexity of ElementaryPath.
8 Evaluation of compositions

As defined earlier, a potential composition is an instance of the composition graph $G = (X, V)$. Each composition can be seen as a collection of individual Web services. The evaluation provided by the UDDI registry are relative to these individual Web services. However, to evaluate and compare the different potential compositions, it is required to define a set of rules to combine the partial evaluations (i.e. in respect to individual Web services) to obtain partial evaluations that apply to the whole composition.

To compute the partial evaluations $g_j(k_i)$ ($j = 1, \ldots, m$) of the different compositions $k_i$ ($i = 1, \ldots, n$), we need to define a set of $m$ aggregation operators $\Phi_1, \Phi_2, \ldots, \Phi_m$, one for each evaluation criterion. The partial evaluation of a composition $k_i$ on criterion $g_j$, $g_j(k_i)$, is computed as follows. It consists in applying a bottom-up scan on graph $G_i = (X_i, V_i)$ and to apply the aggregation operator $\Phi_j$ on each node. Algorithm PartialEvaluation below implements this idea. It runs on $O(r^2)$ where $r = |X|$ is the number of nodes in the composition graph.

The valuation, in respect to criterion $g_j$, of a node $x \in X_i$, denoted $v_j(x)$, is computed as follows:

$$v_j(x) = \Phi_j[g_j(x), \Omega(\Gamma^+(x))]$$

Recall that $\Gamma^+(x)$ is the set of successors of node $x$. The operator $\Omega$ involves nodes on the same level and may be any aggregation operator such as sum, product, max, min, average, etc. The operator $\Phi_j$ implies nodes on different levels and vary according to the BPEL constructors (see Section 7) associated with node $x$. It may be the sum, product, max, min, or average.

---

**Algorithm PartialEvaluation**

**INPUT:** $k_i = G_i(X_i, V_i)$: composition

**INPUT:** $\Phi_j$: aggregation operators

**OUTPUT:** $g_j(k_i)$: partial evaluation of $k_i$ on $g_j$

$L_r \leftarrow \{ s \in X_i : \Gamma^+(s) = \emptyset \}$

$Z \leftarrow \emptyset$

**WHILE** $Z \neq X_i$

**FOR** each $x \in L_r$

$v_j(x) \leftarrow \Phi_j[g_j(x), \Omega(\Gamma^+(x))]$

$Z \leftarrow Z \cup \{x\}$

**END ** FOR

$L_r \leftarrow \{ s \in X_i : v_j(w) \text{ is computed } \forall w \in \Gamma^+(s) \}$

**END ** WHILE

$g_j(k_i) \leftarrow v_j(s)$ where $s$ is the root of $G_i$

---

It is important to note that when the criterion is ordinal, it is not possible to use the probability associated with the branches of a $<\text{switch}>$ constructor. To avoid this problem, we may use one of the following rules (other rules may also apply):
- ignore the probabilities and proceed as with the `<flow>` BPEL constructor;
- use the partial evaluation associated with the most probable branch;
- use the majority rule (when there are at least three branches);
- use the intermediate level between the partial evaluations associated with most probable branch and least probable branch;
- use the intermediate level between the highest partial evaluation and the lowest partial evaluation.

In the following, we provide the proposed formula for computing \( v_j(x) \) \((j = 1, \ldots, 4)\) for response time, availability, cost and security evaluation criteria, denoted \(g_1, g_2, g_3\) and \(g_4\), respectively. Evaluation criteria \(g_1\) and \(g_3\) are to be minimized while criteria \(g_2\) and \(g_4\) are to be maximized. The three first criteria are cardinal. The latter is an ordinal one.

First, we mention that the following formula apply for non-leaf nodes, i.e., \(x \in X_i\) such that \(\Gamma^+(x) \neq \emptyset\). For leaf nodes, i.e. \(x \in X_i\) such that \(\Gamma^+(x) = \emptyset\), the partial evaluation on a criterion \(g_j\) is simply \(v_j(x) = g_j(x)\).

**Response time \((g_1)\)** The response time of a non-leaf node \(x\) is computed as follows:

\[
v_1(x) = g_1(x) + \max\{v_1(y) : y \in \Gamma^+(x)\}
\]

or

\[
v_1(x) = g_1(x) + \sum_{y \in \Gamma^+(x)} \pi(x, y) \cdot v_1(y)
\]

Equation (1) applies for the `<flow>` or the sequential BPEL constructors. Equation (2) applies when the constructor `<switch>` is used. Here: \(\Phi_1\) is the sum and \(\Omega\) is the max (for Equation (1)) or the sum (for Equation (2)).

**Availability \((g_2)\)** For the availability, two formulae may be applied:

\[
v_2(x) = g_2(x) \cdot \prod_{y \in \Gamma^+(x)} v_2(y)
\]

or

\[
v_2(x) = g_2(x) \cdot \sum_{y \in \Gamma^+(x)} \pi(x, y) \cdot v_2(y)
\]

Equation (3) applies for the `<flow>` BPEL or the sequential constructors. Equation (4) applies when the constructor `<switch>` is used. Here: \(\Phi_2\) is the product and \(\Omega\) is the product (for Equation (3)) or the sum (for Equation (4)).
Cost \((g_3)\) For cost criterion, two formula may be used:

\[
v_3(x) = g_3(x) + \sum_{y \in \Gamma^+(x)} v_3(y)
\]

(5)

or

\[
v_3(x) = g_3(x) + \sum_{y \in \Gamma^+(x)} \pi(x, y) \cdot v_3(y)
\]

(6)

Equation (5) applies for the <flow> or the sequential BPEL constructors. Equation (6) applies when the constructor <switch> is used. Here, the \(\sum\) operator is used for both \(\Phi_3\) and \(\Omega\).

Security \((g_4)\) Finally, for security criterion, we have:

\[
v_4(x) = \min\{g_4(x), \min_{y \in \Gamma^+(x)} \{v_4(y)\}\}
\]

(7)

Here, both \(\Phi_4\) and \(\Omega\) are the \(\min\) operator. Recall that security criterion is an ordinal one. Equation (7) applies when the <flow> BPEL constructor is used. When the constructor <switch> is used, one of the rules mentioned above is used.

These different equations are illustrated through Example 2 that follows.

Example 2. For better illustration of the previous formula, consider again the composition graph of Figure 10 and the three compositions \(k_{49}\), \(k_{112}\) and \(k_{185}\) given in Figure 11. Suppose that \(p_1 = 0.4\) and \(p_2 = 0.6\). The objective is to show the computing of partial evaluations of \(k_{49}\), \(k_{112}\) and \(k_{185}\) in respect to four evaluation criteria: Response time \((g_1)\), Availability \((g_2)\), Cost \((g_3)\) and Security \((g_4)\). The evaluation of the providers of Web services \(S_1\) to \(S_6\) in respect to \(g_1\), \(g_2\), \(g_3\) and \(g_4\) are given in Tables 2 to 7, respectively. Recall that evaluation criteria \(g_1\) and \(g_3\) are to be minimized while criteria \(g_2\) and \(g_4\) are to be maximized. Recall also that the three first criteria are cardinal. The latter is ordinal for which the following five-levels scale is used: 1: “very low”, 2: “low”, 3: “average”, 4: “high”, and 5: “very high”.

<table>
<thead>
<tr>
<th>Response time ((g_1))</th>
<th>Availability ((g_2))</th>
<th>Cost ((g_3))</th>
<th>Security ((g_4))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(s_1^1)</td>
<td>2.0</td>
<td>0.99</td>
<td>0.4</td>
</tr>
<tr>
<td>(s_2^1)</td>
<td>1.7</td>
<td>0.95</td>
<td>0.5</td>
</tr>
<tr>
<td>(s_3^1)</td>
<td>1.6</td>
<td>0.80</td>
<td>0.7</td>
</tr>
<tr>
<td>(s_4^1)</td>
<td>3.4</td>
<td>0.94</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Table 2: Evaluations of providers of Web service \(S_1\)

Illustrating now the computing for partial evaluation of composition \(k_{49}\) in respect to response time criterion \((g_1)\). To compute \(g_1(k_{49})\), Algorithm PartialEvaluation
<table>
<thead>
<tr>
<th>Response time ($g_1$)</th>
<th>Availability ($g_2$)</th>
<th>Cost ($g_3$)</th>
<th>Security ($g_4$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1^3$</td>
<td>2.0</td>
<td>0.99</td>
<td>0.7</td>
</tr>
<tr>
<td>$s_2^3$</td>
<td>3.0</td>
<td>0.89</td>
<td>0.6</td>
</tr>
<tr>
<td>$s_3^3$</td>
<td>2.5</td>
<td>0.82</td>
<td>0.4</td>
</tr>
</tbody>
</table>

Table 3. Evaluations of providers of Web service $S_2$

<table>
<thead>
<tr>
<th>Response time ($g_1$)</th>
<th>Availability ($g_2$)</th>
<th>Cost ($g_3$)</th>
<th>Security ($g_4$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1^4$</td>
<td>2.0</td>
<td>0.85</td>
<td>0.4</td>
</tr>
<tr>
<td>$s_2^4$</td>
<td>1.7</td>
<td>0.84</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Table 4. Evaluations of providers of Web service $S_3$

<table>
<thead>
<tr>
<th>Response time ($g_1$)</th>
<th>Availability ($g_2$)</th>
<th>Cost ($g_3$)</th>
<th>Security ($g_4$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1^5$</td>
<td>1.8</td>
<td>0.89</td>
<td>0.3</td>
</tr>
</tbody>
</table>

Table 5. Evaluations of providers of Web service $S_4$

<table>
<thead>
<tr>
<th>Response time ($g_1$)</th>
<th>Availability ($g_2$)</th>
<th>Cost ($g_3$)</th>
<th>Security ($g_4$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1^6$</td>
<td>3</td>
<td>0.86</td>
<td>0.5</td>
</tr>
<tr>
<td>$s_2^6$</td>
<td>1.5</td>
<td>0.60</td>
<td>0.6</td>
</tr>
<tr>
<td>$s_3^6$</td>
<td>2</td>
<td>0.99</td>
<td>0.8</td>
</tr>
<tr>
<td>$s_4^6$</td>
<td>2.5</td>
<td>0.82</td>
<td>1.2</td>
</tr>
<tr>
<td>$s_5^6$</td>
<td>3</td>
<td>0.90</td>
<td>0.6</td>
</tr>
</tbody>
</table>

Table 6. Evaluations of providers of Web service $S_5$

<table>
<thead>
<tr>
<th>Response time ($g_1$)</th>
<th>Availability ($g_2$)</th>
<th>Cost ($g_3$)</th>
<th>Security ($g_4$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1^5$</td>
<td>3</td>
<td>0.8</td>
<td>0.23</td>
</tr>
<tr>
<td>$s_2^5$</td>
<td>2.5</td>
<td>0.92</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Table 7. Evaluations of providers of Web service $S_6$

is used on $G_{4g} = (X_{4g}, V_{4g})$. Details of computing are given below. Recall that aggregation mechanism $\Phi_1$ associated with $g_1$ is the sum operator.

First, for leaf-node $s_1^5$, we have $v_1(s_1^5) = g_1(s_1^5) = 3.0$. Then, for nodes $s_1^4$ and $s_2^5$, we apply Equation (1):

$v_1(s_1^4) = g_1(s_1^4) + \max\{v_1(s_1^5)\}$

$= 1.8 + 3.0$

$= 4.8.$

$v_1(s_2^5) = g_1(s_2^5) + \max\{v_1(s_1^4)\}$

$= 3.0 + 3.0$

$= 6.0.$
For node $s_3^1$, we apply Equation (2):

$$v_1(s_3^1) = g_1(s_3^1) + (p_1 \cdot v_1(s_4^1) + p_2 \cdot v_1(s_5^1))$$
$$= 2.0 + (0.4 \times 4.8 + 0.6 \times 6)$$
$$= 7.52.$$

For node $s_3^2$, we apply Equation (1):

$$v_1(s_3^2) = g_1(s_3^2) + \max\{v_1(s_6^1)\}$$
$$= 2.5 + 3.0$$
$$= 5.5.$$

For node $s_1^1$, we apply Equation (1):

$$v_1(s_1^1) = g_1(s_1^1) + \max\{v_1(s_4^1), v_1(s_5^1)\}$$
$$= 2.0 + \max\{5.5, 7.52\}$$
$$= 2.0 + 7.52$$
$$= 9.52.$$

The partial evaluation of composition $k_{49}$ on criterion response time is then $g_1(k_{49}) = 9.28$.

Consider now the evaluation of composition $k_{49}$ on security criterion $(g_4)$. For leaf-node $s_6^1$, we have $v_4(s_6^1) = g_4(s_6^1) = 5.0$. For nodes $s_4^1$ and $s_5^1$, we apply Equation (7):

$$v_4(s_4^1) = \min\{g_4(s_4^1), \min\{v_4(s_1^4), v_4(s_5^5)\}\}$$
$$= \min\{5, \min\{5\}\}$$
$$= 5.$$

$$v_4(s_5^5) = \min\{g_4(s_5^5), \min\{v_4(s_1^4), v_4(s_5^5)\}\}$$
$$= \min\{3, \min\{5\}\}$$
$$= 3.$$

For node $s_3^1$, we apply Equation (7). Remark that the security criterion is an ordinal one. Thus, we have used the first rule in the list given earlier, that is, we have ignored the probabilities $p_1$ and $p_2$ and proceed as with the $<\text{flow}>$ BPEL constructor by using the $\min$ operator.

$$v_4(s_3^1) = \min\{g_4(s_3^1), \min\{v_4(s_1^4), v_4(s_5^5)\}\}$$
$$= \min\{4, \min\{5, 3\}\}$$
$$= 3.$$
For node $s_2^3$, we apply Equation (7):

$$v_4(s_2^3) = \min\{g_4(s_2^3), \min\{v_4(s_6^5)\}\}$$
$$= \min\{1, 5\}$$
$$= 1.$$

For node $s_1^1$, we apply Equation (7):

$$v_4(s_1^1) = \min\{g_4(s_1^1), \min\{v_4(s_2^3), v_4(s_3^1)\}\}$$
$$= \min\{1, \min\{1, 3\}\}$$
$$= 1.$$

Finally, the partial evaluation of composition $k_{49}$ on $g_4$ is: $g_4(k_{49}) = 1$

The partial evaluations of compositions $k_{49}$, $k_{112}$ and $k_{185}$, in respect to the four criteria $g_1$, $g_2$, $g_3$ and $g_4$ are summed-up in the performance table of Figure 8.

<table>
<thead>
<tr>
<th>Composition</th>
<th>Response Time ($g_1$)</th>
<th>Availability ($g_2$)</th>
<th>Cost ($g_3$)</th>
<th>Security ($g_4$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_{49}$</td>
<td>9.52</td>
<td>0.39567</td>
<td>2.14</td>
<td>1</td>
</tr>
<tr>
<td>$k_{112}$</td>
<td>8.42</td>
<td>0.48296</td>
<td>2.82</td>
<td>1</td>
</tr>
<tr>
<td>$k_{185}$</td>
<td>10.32</td>
<td>0.48093</td>
<td>2.16</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 8. Partial evaluations of compositions $k_{49}$, $k_{112}$ and $k_{185}$

Algorithm PartialEvaluation permits to evaluate a given composition on a single criterion. Algorithm PerformanceTable below permits to obtain the complete performance table containing the evaluations of all potential compositions $k_i \in K$ in respect to all evaluation criteria $g_j \in F$. PerformanceTable is straightforward. It simply loops on the set of compositions $K$ and on the set of criteria $F$ and call Algorithm PartialEvaluation to compute the partial evaluation of each composition in $K$ in respect to each criterion in $F$. Algorithm PerformanceTable runs on $O(n \times m \times r)$ where $n$ is the number of compositions, $m$ is the number of evaluation criteria and $r$ is the complexity of PartialEvaluation.
Algorithm PerformanceTable

**INPUT:** \( K = \{k_1, k_2, \cdots, k_n\} \): potential compositions
\( \Phi = \{\Phi_1, \Phi_2, \cdots, \Phi_m\} \): aggregation operators

**OUTPUT:** \( g_j(k_i) \ (i = 1, \cdots, n) \ (j = 1, \cdots, m) \):

PerforTable: matrix of \( n \) rows and \( m \) columns

FOR \( i = 1 \) to \( n \)
FOR \( j = 1 \) to \( m \)
   \( \text{PerforTable}(i, m) \leftarrow \text{PartialEvaluation}(G_i(X_i, V_i), \Phi_j) \)
END FOR
END FOR

One important remark to conclude this section is related to the evaluation of individual Web services. In the formula given above, we have supposed that the partial evaluations of individual Web services \( g_j(.) \) is available on the UDDI registry. However, this is not always true because these information are not often specified by the providers.

9 Illustrative example

The proposed framework is being developed. The objective of this section is simply to show the its feasibility. For this purpose we consider the same composition graph example introduced in Section 7 and shown in Figure 10. The objective is to classify the different potential compositions into different ordered categories. For the purpose of this example, four categories have been defined. The profile limits associated with these categories are given in Table 9. As it is shown in this table, the indifference thresholds for all criteria are equal to 0. This means that any difference in the evaluation is taken into account.

<table>
<thead>
<tr>
<th>( g_1 )</th>
<th>( g(b_3) )</th>
<th>( q_3 )</th>
<th>( g(b_2) )</th>
<th>( q_2 )</th>
<th>( g(b_1) )</th>
<th>( q_1 )</th>
<th>( p_1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.295</td>
<td>0.25</td>
<td>9.175</td>
<td>0</td>
<td>0.25</td>
<td>10.045</td>
<td>0</td>
<td>0.25</td>
</tr>
<tr>
<td>0.56553</td>
<td>0.03</td>
<td>0.46119</td>
<td>0</td>
<td>0.03</td>
<td>0.35685</td>
<td>0</td>
<td>0.03</td>
</tr>
<tr>
<td>2.27</td>
<td>0.02</td>
<td>2.76</td>
<td>0</td>
<td>0.02</td>
<td>3.25</td>
<td>0</td>
<td>0.02</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 9. Parameters of profile limits

Next, we suppose that the consumer is not able to provide all the required preference parameters explicitly. Instead, s/he provides the assignment examples given in Table 10, which can be used to infer the different preference parameters. Columns \( cMin(k_i) \) and \( cMax(k_i) \) are, respectively, the lowest and highest
categories to which composition $k_i$ should be assigned. Compositions for which the lowest and highest categories are equal correspond to exact assignments, i.e., only one category is possible. The other assignment examples correspond to the case where a range of assignments is possible. For instance, composition $k_{181}$ in Table 10 must be assigned to $C_1$ while composition $k_{11}$ may be assigned to $C_1$, $C_2$ or $C_3$.

The computational complexity for solving the mathematical program varies in the sense that the mathematical program to resolve may be linear or non-linear. Specifically, with an outranking relation, obtaining a global optimum is not obvious and requires the resolution of a nonlinear mathematical program. One possible solution to overcome this difficulty is to use “partial” inference procedures. Indeed, if the value of some preference parameters can be considered as known, “partial” inference procedures can be applied to infer the other parameters. A partial inference is useful in situations in which the value of some parameters can reasonably be set. If not, it is possible to partition the parameters in sets, and proceed through a sequence of partial inference procedures in which the value of some parameters is fixed. In this paper, the inference procedure was used to infer the cutting level and the weights of the different evaluation criteria. The other (fixed) parameters are provided in Table 11.

<table>
<thead>
<tr>
<th>$k_i$</th>
<th>Description</th>
<th>$g_1(k_i)$</th>
<th>$g_2(k_i)$</th>
<th>$g_3(k_i)$</th>
<th>$g_4(k_i)$</th>
<th>$\text{Min}(k_i)$</th>
<th>$\text{Max}(k_i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_{71}$</td>
<td>$s_1, s_2, s_1, s_2, s_3, s_3$</td>
<td>9.22</td>
<td>0.45940</td>
<td>2.38</td>
<td>1</td>
<td>$C_1$</td>
<td>$C_3$</td>
</tr>
<tr>
<td>$k_{11}$</td>
<td>$s_1, s_1, s_1, s_1, s_2, s_2$</td>
<td>8.12</td>
<td>0.41817</td>
<td>2.68</td>
<td>1</td>
<td>$C_1$</td>
<td>$C_2$</td>
</tr>
<tr>
<td>$k_{10}$</td>
<td>$s_2, s_3, s_1, s_3, s_5, s_3$</td>
<td>8.72</td>
<td>0.50210</td>
<td>2.78</td>
<td>2</td>
<td>$C_3$</td>
<td>$C_4$</td>
</tr>
<tr>
<td>$k_{10}$</td>
<td>$s_2, s_3, s_3, s_5, s_5, s_2$</td>
<td>8.12</td>
<td>0.46067</td>
<td>3.24</td>
<td>2</td>
<td>$C_4$</td>
<td>$C_4$</td>
</tr>
<tr>
<td>$k_{11}$</td>
<td>$s_2, s_3, s_1, s_1, s_5, s_3$</td>
<td>9.12</td>
<td>0.38604</td>
<td>2.74</td>
<td>3</td>
<td>$C_2$</td>
<td>$C_3$</td>
</tr>
<tr>
<td>$k_{13}$</td>
<td>$s_3, s_1, s_2, s_1, s_3, s_2$</td>
<td>7.42</td>
<td>0.40317</td>
<td>3.38</td>
<td>2</td>
<td>$C_2$</td>
<td>$C_3$</td>
</tr>
<tr>
<td>$k_{14}$</td>
<td>$s_3, s_2, s_3, s_1, s_3, s_2$</td>
<td>7.72</td>
<td>0.36676</td>
<td>3.18</td>
<td>2</td>
<td>$C_2$</td>
<td>$C_4$</td>
</tr>
<tr>
<td>$k_{15}$</td>
<td>$s_3, s_2, s_3, s_3, s_5, s_2$</td>
<td>8.82</td>
<td>0.34296</td>
<td>2.74</td>
<td>2</td>
<td>$C_3$</td>
<td>$C_3$</td>
</tr>
<tr>
<td>$k_{15}$</td>
<td>$s_3, s_1, s_2, s_1, s_5, s_3$</td>
<td>8.12</td>
<td>0.44145</td>
<td>2.68</td>
<td>1</td>
<td>$C_3$</td>
<td>$C_1$</td>
</tr>
<tr>
<td>$k_{16}$</td>
<td>$s_3, s_2, s_3, s_5, s_5, s_2$</td>
<td>9.52</td>
<td>0.56508</td>
<td>2.8</td>
<td>2</td>
<td>$C_3$</td>
<td>$C_2$</td>
</tr>
<tr>
<td>$k_{17}$</td>
<td>$s_3, s_3, s_2, s_1, s_2, s_3$</td>
<td>10.12</td>
<td>0.53294</td>
<td>2.68</td>
<td>2</td>
<td>$C_3$</td>
<td>$C_3$</td>
</tr>
<tr>
<td>$k_{18}$</td>
<td>$s_3, s_3, s_1, s_1, s_3, s_2$</td>
<td>10.42</td>
<td>0.48356</td>
<td>2.32</td>
<td>1</td>
<td>$C_1$</td>
<td>$C_2$</td>
</tr>
</tbody>
</table>

Table 10. Assignment examples

The output of the inference procedure is given in Table 12. According to Table 12, “Response time” ($g_1$) and “Availability” ($g_2$) evaluation criteria have the same relative importance with weights $w_1 = w_2 = 0.32502$. Equally, “Cost”

<table>
<thead>
<tr>
<th>Preference parameter</th>
<th>Response time ($g_1$)</th>
<th>Availability ($g_2$)</th>
<th>Cost ($g_3$)</th>
<th>Security ($g_4$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indifference threshold ($q_i$)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Preference threshold ($p_i$)</td>
<td>0.25</td>
<td>0.03</td>
<td>0.02</td>
<td>1</td>
</tr>
<tr>
<td>Veto threshold ($v_i$)</td>
<td>not used</td>
<td>not used</td>
<td>not used</td>
<td>not used</td>
</tr>
</tbody>
</table>

Table 11. Fixed Preference parameters
(g_3) and “Security” (g_4) evaluation criteria are of equal importance with w_3 = w_4 = 0.17498. The value of the cutting level is 0.57507.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>( \lambda )</th>
<th>( w_1 )</th>
<th>( w_2 )</th>
<th>( w_3 )</th>
<th>( w_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inferred value</td>
<td>0.57507</td>
<td>0.32502</td>
<td>0.32502</td>
<td>0.17498</td>
<td>0.17498</td>
</tr>
</tbody>
</table>

Table 12. Inferred preference parameters

The obtained preference parameters are then used to apply ELECTRE TRI. The result of classification for assignment examples is summed up in Table 13. The result for all the compositions is included on the last three columns of Table 15 in Appendix A. The list of best compositions (i.e., those belonging in \( C_1 \)) is shown in Table 14. The last column in Table 14 shows the order of best compositions obtained by a weighted sum of \( g_1 \), \( g_2 \), \( g_3 \) and \( g_4 \):

\[
g(k_i) = -w_1 \cdot g_1(k_i) + w_2 \cdot g_2(k_i) - w_3 \cdot g_3(k_i) + w_4 \cdot g_4(k_i)
\]  

(8)

In Equation 8 we supposed that criterion \( g_4 \) (security) is cardinal and \( w_1 \), \( w_2 \), \( w_3 \) and \( w_4 \) are the relative importances obtained by inference. The ten top compositions using the weighted sum are: \( k_{114} \ k_{166} \ k_{174} \ k_{94} \ k_{126} \ k_{146} \ k_{154} \ k_{134} \ k_{116} \ k_{104} \). It is easy to see that none of best compositions (obtained by ELECTRE TRI) are among these ten top compositions.

<table>
<thead>
<tr>
<th>( k_i )</th>
<th>Description</th>
<th>Worst category</th>
<th>Inferred category</th>
<th>Best category</th>
</tr>
</thead>
<tbody>
<tr>
<td>( k_{11} )</td>
<td>( s_1, s_2, s_2, s_2, s_2, s_2 )</td>
<td>( C_2 )</td>
<td>( C_3 )</td>
<td>( C_5 )</td>
</tr>
<tr>
<td>( k_{44} )</td>
<td>( s_1, s_3, s_1, s_1, s_2, s_2 )</td>
<td>( C_2 )</td>
<td>( C_2 )</td>
<td>( C_2 )</td>
</tr>
<tr>
<td>( k_{110} )</td>
<td>( s_2, s_3, s_1, s_1, s_5, s_2 )</td>
<td>( C_3 )</td>
<td>( C_3 )</td>
<td>( C_3 )</td>
</tr>
<tr>
<td>( k_{116} )</td>
<td>( s_2, s_3, s_2, s_1, s_2, s_2 )</td>
<td>( C_3 )</td>
<td>( C_3 )</td>
<td>( C_3 )</td>
</tr>
<tr>
<td>( k_{129} )</td>
<td>( s_2, s_3, s_1, s_1, s_5, s_1 )</td>
<td>( C_2 )</td>
<td>( C_3 )</td>
<td>( C_3 )</td>
</tr>
<tr>
<td>( k_{134} )</td>
<td>( s_3, s_3, s_2, s_1, s_2, s_2 )</td>
<td>( C_2 )</td>
<td>( C_2 )</td>
<td>( C_2 )</td>
</tr>
<tr>
<td>( k_{144} )</td>
<td>( s_5, s_5, s_5, s_5, s_5, s_2 )</td>
<td>( C_2 )</td>
<td>( C_2 )</td>
<td>( C_2 )</td>
</tr>
<tr>
<td>( k_{157} )</td>
<td>( s_2, s_2, s_1, s_1, s_5, s_1 )</td>
<td>( C_2 )</td>
<td>( C_2 )</td>
<td>( C_2 )</td>
</tr>
<tr>
<td>( k_{180} )</td>
<td>( s_4, s_1, s_1, s_1, s_1, s_1 )</td>
<td>( C_1 )</td>
<td>( C_1 )</td>
<td>( C_1 )</td>
</tr>
<tr>
<td>( k_{210} )</td>
<td>( s_4, s_3, s_2, s_1, s_3, s_2 )</td>
<td>( C_2 )</td>
<td>( C_2 )</td>
<td>( C_2 )</td>
</tr>
<tr>
<td>( k_{220} )</td>
<td>( s_4, s_3, s_1, s_1, s_3, s_2 )</td>
<td>( C_2 )</td>
<td>( C_2 )</td>
<td>( C_2 )</td>
</tr>
</tbody>
</table>

Table 13. Result of classification for the assignment examples

10 Discussion

In this section we discuss the computational behavior of the proposed framework. Indeed, the solution proposed to define the composite Web services may lead to a large set of potential compositions. To guarantee a minimum level of efficiency, it is required that the time to construct, evaluate and compare the
different potential compositions does not exceed the response time of the worst compositions. Naturally, the worst composition may be a fictive one. The worst composition \( k_w \) is defined as the one such that \( \forall j \in F: \)

\[- g_j(k_w) \leq \min_{k \in K} g_j(k) \text{ if preference increases on } g_j, \]

\[- g_j(k_w) \geq \max_{k \in K} g_j(k) \text{ if preference decreases on } g_j. \]

Another possible solution follows. Let \( \mathcal{T}(k) \) be the response time of composition \( k \) and \( \tau \) the time required to construct, evaluate and compare potential compositions. The number \( \tau \) may be estimated based on the complexity of the different algorithms and on the average execution time of basic operations. Then, the following proposition should be verified.

**Proposition 1** If \( \mathcal{T}(k_w) > \tau \), then use MEC. Otherwise, use conventional evaluation.

This rule may be included in the registry and applied to orient the system towards conventional evaluation or towards multicriteria evaluation.

In addition to this proposition, the framework may be further enhanced through some basic notions in multicriteria analysis. First, the number of potential compositions may be largely reduced by using the pre-analyse of dominance. The pre-analyse of dominance permits to eliminate from consideration all the compositions which are dominated by at least another one. Let \( k_i \) and \( k_l \) be two composite Web services from \( K \). Suppose that preference is increasing on all criteria (this is not restrictive). Then, composite Web service \( k_i \) dominates Composite Web service \( k_l \) in respect to \( F \), denoted \( k_i \Delta k_l \), if and only if \( g_j(k_i) \geq g_j(k_l) \forall j \in F \), with at least one strict inequality. Only non dominated composite Web services are considered in the multicriteria evaluation step.

Another solution consists in using some elementary multicriteria methods. Two typical elementary multicriteria methods are the conjonctive and disjonctive decision rules. In the conjonctive method, we define, for each criterion \( g_j \), a minimal satisfaction level \( \hat{g}_j \). A composition \( k_i \) is acceptable if and only if:

\[ g_j(k_i) \geq \hat{g}_j, \forall j \in F \]

A more drastic choice consists in eliminating any composition having performance vector \((g_1(k_i), g_2(k_i), \ldots, g_m(k_i))\) dominated by the vector \((\hat{g}_1, \hat{g}_2, \ldots, \hat{g}_m)\).
The disjonctive method is similar to the previous one but a composition is acceptable if it exceed at least one satisfaction level:

$$\exists j \in F : g_j(k_i) > \hat{g}_j$$

More complex elementary methods as the elimination by aspect (EBA) and lexicographic elimination ones may also be used. With EBA method, a set of satisfaction levels are defined and applied progressively. In each step, the composition that fails to verify these satisfaction levels is eliminated. The procedure is repeated for all the remaining compositions using the next criterion. The lexicographic elimination method supposes that an order on the evaluation criteria is established and that the \((k-1)\) next criteria may be ignored if the first \(k\) criteria are sufficient to make a decision.

11 Conclusion

We have proposed a framework for composite Web services selection based on multicriteria evaluation. The framework extends the conventional Web services architecture by adding a new Multicriteria Evaluation Component (MEC) devoted to multicriteria evaluation. This additional component takes as input a set of composite Web services and a set of evaluation criteria. The output is a set of recommended composite Web services. We also proposed solutions to construct and evaluate the different potential compositions. We also discussed the computational behavior of the framework and proposed some solutions to reduce the complexity of the solution. To show the feasibility of our proposal, an illustrative example is included in the paper. A prototype implementing the proposed framework is under development.

Currently, we are concerned with the finalization of the being developed prototype. Aside this, there are several directions for future research. A first point to investigate is related to the extension of the framework to support dynamic composition. The basic change concerns the construction of the potential compositions and their evaluations.

References


A Presentation of ELECTRE TRI

ELECTRE TRI is a multicriteria sorting method used to assign alternatives to predefined ordered categories. The assignment of an alternative $a$ results from the comparison of $a$ with the profiles defining the limits of the categories. Let $A$ denote the set of alternatives to be assigned. Let $F$ denote the set of the indices of the criteria $g_1, g_2, \cdots, g_m$ ($F = 1, 2, \cdots, m$), $k_j$ the importance coefficient of the criterion $g_j$, $B$ the set of indices of the profiles defining $p + 1$ categories ($B = 1, 2, \cdots, p$), $b_h$ being the upper limit of category $C_h$ and the lower limit of category $C_{h+1}$, $h = 1, 2, \cdots, p$.

ELECTRE TRI builds a fuzzy outranking relation $S$ whose meaning is “at least as good as”. Preferences on each criterion are defined through pseudo-criteria. The threshold $q_j(b_h)$ represents the largest difference $g_j(a) - g_j(b_h)$ preserving an indifference between $a$ and $b_h$ in respect to criterion $g_j$. The threshold $p_j(b_h)$ represents the smallest difference $g_j(a) - g_j(b_h)$ compatible with a preference in favor of $a$ in respect to criterion $g_j$. Thus, the limits of categories are defined in terms of profiles $b_h$, $h \in B$; each one is delimited by two imprecision zones.

To validate the proposition $aSb_h$, two conditions must hold:

- **Concordance**: An outranking $aSb_h$ is accepted only if a “sufficient” majority of criteria are in favor of this proposition.
- **Non-discordance**: When the concordance holds, none of the minority of criteria shows an “important” opposition to $aSb_h$.

Besides the intra-criterion preferential information, represented by the indifference and preference thresholds, $q_j(b_h)$ and $p_j(b_h)$, the construction of $S$ also makes use of two types of inter-criterion preferential information:

- the set of weight-importance coefficients $(w_1, w_2, \cdots, w_m)$ is used in the concordance test when computing the relative importance of the coalitions of criteria being in favor of the assertion $aSb_h$.
- the set of veto thresholds $(v_1(b_h), v_2(b_h), \cdots, v_m(b_h))$; $h \in B$, is used in the discordance test; $v_j(b_h)$ represents the smallest difference $g_j(b_h) - g_j(a)$ incompatible with the assertion $aSb_h$.

As the assignment of alternatives to categories does not result directly from the relation $S$, an exploitation phase is necessary; it requires the relation $S$ to be “defuzzyfied” using a so-called $\lambda$-cut: the assertion $aSb_h$ is considered to be valid if the credibility index of the fuzzy outranking relation is greater than a “cutting level” $\lambda$ such that $\lambda \in [0.5, 1]$. ELECTRE TRI constructs an indices $\sigma(a, b_h) \in [0, 1]$ representing the credibility of the proposition $aSb_h$, $a \in A$, $h \in B$. The proposition $aSb_h$ holds if $\sigma(a, b_h) \geq \lambda$. The indices $(a, b_h)$ is defined using algorithm $\text{SIGMA}$ given in Appendix B.

Two assignment procedures are available: optimistic and pessimistic. Their role being to analyze the way in which an alternative $a$ compares to the profiles so as to determine the category to which $a$ should be assigned. The result of
these two assignment procedures differs when the alternative \(a\) is incomparable with at least one profile \(b_h\)

- Pessimistic procedure:
  - Compare \(a\) successively to \(b_i\); \(i = p, p - 1, \ldots, 0\).
  - Let \(b_h\) the first profile such that \(aSb_h\), then assign \(a\) to category \(C_{h+1}\).

- Optimistic procedure:
  - Compare \(a\) successively to \(b_i\); \(i = 1, 2, \ldots, p\).
  - Let \(b_h\) the first profile such that \(b_hSa\), then assign \(a\) to category \(C_h\).

B Algorithm SIGMA

\[
\text{Algorithm SIGMA}
\]

**INPUT:**
- \(g_1(a), \ldots, g_m(a)\): performances of alternative \(a\)
- \(g_1(b_h), \ldots, g_m(b_h)\): performances of profile \(b_h\)
- \(q_j(b_h)\) \((j = 1, \ldots, m)\): indifference thresholds of \(b_h\)
- \(p_j(b_h)\) \((j = 1, \ldots, m)\): preference thresholds of \(b_h\)
- \(v_j(b_h)\) \((j = 1, \ldots, m)\): veto thresholds of \(b_h\)
- \(w_j\) \((j = 1, \ldots, m)\): evaluation criteria weights
- \(\lambda\): cutting level

**OUTPUT:**
- \(\sigma(a, b_h)\): credibility indices of assertion \(aSb_h\)

**BEGIN**
1. Compute partial concordance indices \(S_j(a, b_h), \forall j \in F\):

\[
S_j(a, b_h) = \begin{cases} 0, & \text{if } g_j(b_h) - g_j(a) \geq p_j(b_h) \\ 1, & \text{if } g_j(b_h) - g_j(a) \leq q_j(b_h) \\ \frac{p_j(b_h) - g_j(a) + q_j(b_h)}{g_j(b_h) - g_j(a)} & \text{otherwise.} \end{cases}
\]

2. Compute global concordance indice \(S(a, b_h)\):

\[
S(a, b_h) = \sum_{j \in F} k_j \cdot S_j(a, b_h)
\]

3. Compute partial discordance indices \(\text{ND}_j(a, b_h), \forall j \in F\):

\[
\text{ND}_j(a, b_h) = \begin{cases} 0, & \text{if } g_j(a) \leq g_j(b_h) + p_j(b_h) \\ 1, & \text{if } g_j(a) \geq g_j(b_h) + v_j(b_h) \\ \frac{1 - g_j(b_h) + p_j(b_h)}{g_j(a) - g_j(b_h) + v_j(b_h)} & \text{otherwise.} \end{cases}
\]

4. Compute the global discordance indice \(\text{ND}(a, b_h)\):

\[
\text{ND}(a, b_h) = \prod_{j \in F'} \frac{1 - \text{ND}_j(a, b_h)}{1 - S(a, b_h)}
\]

with \(F' = \{ j \in F : \text{ND}_j(a, b_h) > S(a, b_h) \}\)

5. Compute credibility indice \(\sigma(a, b_h)\):

\[
\sigma(a, b_h) = S(a, b_h) \cdot \text{ND}(a, b_h)
\]

**END**

C Performance table and final classification
<table>
<thead>
<tr>
<th>k</th>
<th>( n )</th>
<th>( \rho )</th>
<th>( \sigma )</th>
<th>( \text{Mean val} )</th>
<th>( \text{Inferred cat.} )</th>
<th>( \text{Mean val} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>k1</td>
<td>8.15</td>
<td>0.4952</td>
<td>2.46</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>k2</td>
<td>8.15</td>
<td>0.4952</td>
<td>2.46</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>k3</td>
<td>8.15</td>
<td>0.4952</td>
<td>2.46</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>k4</td>
<td>8.15</td>
<td>0.4952</td>
<td>2.46</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>k5</td>
<td>8.15</td>
<td>0.4952</td>
<td>2.46</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>k6</td>
<td>8.15</td>
<td>0.4952</td>
<td>2.46</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>k7</td>
<td>8.15</td>
<td>0.4952</td>
<td>2.46</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>k8</td>
<td>8.15</td>
<td>0.4952</td>
<td>2.46</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>k9</td>
<td>8.15</td>
<td>0.4952</td>
<td>2.46</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>k10</td>
<td>8.15</td>
<td>0.4952</td>
<td>2.46</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>k11</td>
<td>8.15</td>
<td>0.4952</td>
<td>2.46</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>k12</td>
<td>8.15</td>
<td>0.4952</td>
<td>2.46</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>k13</td>
<td>8.15</td>
<td>0.4952</td>
<td>2.46</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>k14</td>
<td>8.15</td>
<td>0.4952</td>
<td>2.46</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>k15</td>
<td>8.15</td>
<td>0.4952</td>
<td>2.46</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>k16</td>
<td>8.15</td>
<td>0.4952</td>
<td>2.46</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>k17</td>
<td>8.15</td>
<td>0.4952</td>
<td>2.46</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>k18</td>
<td>8.15</td>
<td>0.4952</td>
<td>2.46</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>k19</td>
<td>8.15</td>
<td>0.4952</td>
<td>2.46</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>k20</td>
<td>8.15</td>
<td>0.4952</td>
<td>2.46</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>k21</td>
<td>8.15</td>
<td>0.4952</td>
<td>2.46</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>k22</td>
<td>8.15</td>
<td>0.4952</td>
<td>2.46</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>k23</td>
<td>8.15</td>
<td>0.4952</td>
<td>2.46</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>k24</td>
<td>8.15</td>
<td>0.4952</td>
<td>2.46</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>k25</td>
<td>8.15</td>
<td>0.4952</td>
<td>2.46</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>k26</td>
<td>8.15</td>
<td>0.4952</td>
<td>2.46</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>k27</td>
<td>8.15</td>
<td>0.4952</td>
<td>2.46</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>k28</td>
<td>8.15</td>
<td>0.4952</td>
<td>2.46</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>k29</td>
<td>8.15</td>
<td>0.4952</td>
<td>2.46</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>k30</td>
<td>8.15</td>
<td>0.4952</td>
<td>2.46</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 15: Partial evaluations of compositions and results (continued on the next page)
<table>
<thead>
<tr>
<th>k</th>
<th>Inferred cat.</th>
<th>Best cat.</th>
<th>Worst cat.</th>
</tr>
</thead>
<tbody>
<tr>
<td>124</td>
<td>184</td>
<td>k</td>
<td>k</td>
</tr>
<tr>
<td>133</td>
<td>134</td>
<td>k</td>
<td>k</td>
</tr>
<tr>
<td>144</td>
<td>184</td>
<td>k</td>
<td>k</td>
</tr>
<tr>
<td>150</td>
<td>152</td>
<td>k</td>
<td>k</td>
</tr>
<tr>
<td>155</td>
<td>156</td>
<td>k</td>
<td>k</td>
</tr>
<tr>
<td>159</td>
<td>162</td>
<td>k</td>
<td>k</td>
</tr>
<tr>
<td>164</td>
<td>169</td>
<td>k</td>
<td>k</td>
</tr>
<tr>
<td>171</td>
<td>172</td>
<td>k</td>
<td>k</td>
</tr>
<tr>
<td>173</td>
<td>174</td>
<td>k</td>
<td>k</td>
</tr>
<tr>
<td>178</td>
<td>182</td>
<td>k</td>
<td>k</td>
</tr>
<tr>
<td>189</td>
<td>190</td>
<td>k</td>
<td>k</td>
</tr>
<tr>
<td>193</td>
<td>194</td>
<td>k</td>
<td>k</td>
</tr>
<tr>
<td>199</td>
<td>201</td>
<td>k</td>
<td>k</td>
</tr>
<tr>
<td>202</td>
<td>205</td>
<td>k</td>
<td>k</td>
</tr>
<tr>
<td>207</td>
<td>208</td>
<td>k</td>
<td>k</td>
</tr>
<tr>
<td>210</td>
<td>213</td>
<td>k</td>
<td>k</td>
</tr>
<tr>
<td>214</td>
<td>215</td>
<td>k</td>
<td>k</td>
</tr>
<tr>
<td>218</td>
<td>222</td>
<td>k</td>
<td>k</td>
</tr>
<tr>
<td>226</td>
<td>227</td>
<td>k</td>
<td>k</td>
</tr>
<tr>
<td>228</td>
<td>229</td>
<td>k</td>
<td>k</td>
</tr>
<tr>
<td>230</td>
<td>232</td>
<td>k</td>
<td>k</td>
</tr>
<tr>
<td>237</td>
<td>238</td>
<td>k</td>
<td>k</td>
</tr>
</tbody>
</table>

Table 15. Partial evaluations of compositions and results.