RESEARCH ARTICLE OPEN ACCESS

A Case Elaboration Methodology for a Semantic Web Service Discovery System Based on CBR

Ibrahim El Bitar*, Zouhair Bazzal**, Fatima-Zahra Belouadha***
*Applied Mathematics Dept, Faculty of Sciences, Lebanese University, Lebanon
** Computer and Communication Engineering Dept, School of Engineering, Lebanese International University, Lebanon
*** Computer Engineering Dept, Mohammadia School of Engineering, Mohammed V University, Morocco

ABSTRACT
The Case Based Reasoning is a paradigm of intelligent reasoning which consists on reusing results of previously solved problems (Source Cases) to solve new problems (Target Cases). It has been formalized as a five-step process consisting of: "Elaboration", "Retrieve", "Reuse", "Revise" and "Retain". In this paper we focus on the first phase of the CBR cycle with all of the required modeling to formalize a Case in our CBR-based system for semantic Web service discovery (CBR4WSD). This phase consists in formalizing the problem description and its structuring before launching the "Retrieve" phase and select the most appropriate Source Cases from the Case Base. We identify a set of basic descriptors to formalize Cases handled in our CBR4WSD system. In this conduct and in accordance with CBR policies, we put forward our Case representation model.
Keywords: CBR, case elaboration, Web Services, functional properties, non-functional properties

I. INTRODUCTION

Case Based Reasoning (CBR) is a paradigm for analogical reasoning, used to solve new problems based on the solutions of similar past problems. It has been formalized as a five-step process consisting of: «Elaboration», «Retrieve», «Reuse», «Revise» and «Retain».

In this paper we focus on the first phase of the CBR cycle with all of the required modeling to formalize a Case in our CBR-based system for semantic Web service discovery (CBR4WSD). This phase consists in formalizing the problem description and its structuring before launching the «Retrieve» phase and select the most appropriate Source Cases from the Case Base. We identify a set of basic descriptors to formalize Cases handled in our CBR4WSD system. In this conduct and in accordance with CBR policies, we put forward our Case representation model.

The rest of this paper is organized as follows. Section 2 presents the definition of a Case and its notation. Section 3 outlines the needs' extraction to describe a Target Case (Web service query) in our CBR4WSD system by identifying its functional and non-functional descriptors. Section 4 shows the representation of the solution part of a Case handled in CBR4WSD system. In section 5 we discuss the tasks included in the Target Case Elaboration phase and the encountered knowledge imperfection problems. Section 6 concludes this work and emerges a synthesis.

II. CASE DEFINITION AND NOTATION

We recall that in the CBR terminology, a Case represents different types of knowledge that can be stored in different formats of representation. A Case consists of a problem «pb » and its solution «sol (pb) ». The Case is then noted: Case = (pb, sol(bp)).

A Source Case is a Case that serves as an inspiration to solve a new problem called the Target Case. The coding is as follows:

- Source Case = (Source, sol (Source))
- Target Case = (Target, sol (Target))

The Cases are described by a set of descriptors which depend on the application domain. A descriptor is a knowledge that allows us to describe the problem. The descriptor "d" is characterized by a pair (a, v) [1], where:

- "a" is an attribute defined by a name,
- "v" is the value that is associated with it.

An attribute is a characteristic of the application domain that can be numeric or symbolic. We describe the Case as the following:

- **Source Case:**

Source = {d_1^S .... d_n^S} where d_i^S is a descriptor of the Source problem.

Sol (Source) = {D_1^S ... D_n^S} where D_i^S is a descriptor of the Source solution

- **Target Case:**

Target = {d_1^C ... d_n^C} where d_i^C is a descriptor of the Target problem.

Sol (Target) = {D_1^C ... D_n^C} where D_i^C is a descriptor of the Target solution.

Subsequently, we use these definitions to represent Cases in our CBR4WSD system, which is dedicated to the automatic discovery of Web services in response to the query of a client requesting a service with specific needs.

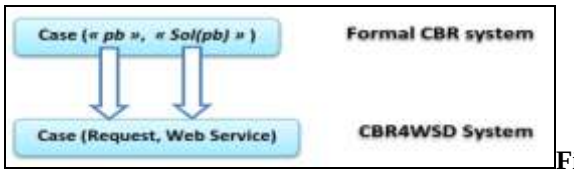

Fig.1. Components of a Case in CBR4WSD system.

Thus, the « pb » part of the Case reflects the client's query and will describe its characteristics in terms of functional and non-functional needs. Similarly, the « sol(pb) » part should identify relevant and sufficient information for the invocation of the required service (Fig.1).

III. WEB SERVICE DESCRIPTION ELEMENTS IN THE CBR4WSD SYSTEM

Web services are usually described using WSDL, the W3C standard. However, the lack of semantics in WSDL prevents the automatic discovery of Web services [2]. For this reason, a series of models incorporating semantics in the description of their Web services have been proposed. Among these non-standardized works, some consider the description of functional requirements [3, 4, 5, 6] and others consider the description of non-functional requirements [7, 8, 9]. These non-standardized description models have been used in many CBR-based approaches dedicated for web service discovery and composition [10, 11, 12, 13, 14, 15]. Nonetheless, these approaches mainly focus on the Web service functional properties.

The results of the study we previously conducted on these approaches have leaded us to propose a semantic description model aligned with W3C standards for Web service expressive description [16]. Also, we were asked to formalize our Cases in accordance with this model and its standards.

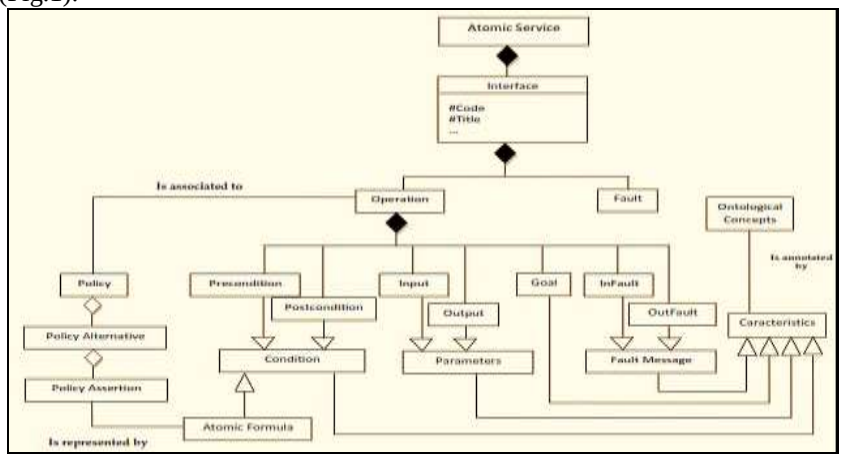

Fig. 2. Class-diagram representing an atomic Web service handled in CBR4WSD system.

We call « Atomic service » an elementary Web service grouping a set of operations. These operations are described using the standard parameters Input, Output, InFault and outfault to which we have added the parameters Goal, Precondition and Postcondition.

The « Characteristic » class generalizes these parameters. It can be semantically annotated using the association « Is annotated by » which connects the characteristic to a « Semantic Concept » in accordance with the principle of semantic annotation of SAWSDL.

The « Condition » class generalizes « Precondition », «Postcondition » and the « Atomic Formula ». According to WS-Policy, the non-functional aspects are expressed by the « Policy » class. A policy is described by a set of « Policy Alternative ». Each « Policy Alternative » includes a set of assertions described by the « Policy Assertion » class.

In order to describe the Web service functional and non-functional aspects, we have presented in this section an extended and enhanced service description. In the following section, we proceed by defining the set of Case descriptors (*pb, sol(pb)*) in accordance with the standards used in our Web service description model.

3.1. Representation of the problem part of a Case handled in CBR4WSD

The Case problem part « (pb) » reflects the client's query seeking a specific operation of a given service.

Our representation of the problem part of a Case handled in CBR4WSD builds on our definition of the Web services discovery mechanism as *"the act of* ***locating*** *a machine treatable description, of a previously unknown Web service describing some* ***functional*** *and* ***non-functional requirements****"*.

We consider these functional and non-functional requirements in the formalization of the Case which builds upon our enriched Web services description model. Thus, in the problem part, we distinguish the functional properties (FP) from the non-functional properties (NFP). Our Case has a functional part and a non-functional one in its problem space, hence the notation: *pb = (FP, NFP*).

A. Functional descriptors of the Case problem part

The functional properties expressed in the Case *« (pb) »* part will be represented by the attributes «Goal, Input, Output, Precondition and Postconditon » relating to an operation of a SAWSDL service. Thus, the descriptors of this part are illustrated in Table 1.

	Number and name of descriptor	Definition
Functional Descriptors (mandatory)	ds1 : « Goal »	Purpose of the required operation of the SAWSDL service.
	ds2 : <Inputs>	List of input parameters of the required operation.
	ds3 : <Outputs>	List of output parameters of the required operation.
Functional Descriptors (optional)	ds4 : <Preconditions>	List of preconditions imposed on the required operation.
	ds5 : <Postconditions>	List of postconditions imposed on the required operation.

Table 1. Functional descriptors of the Case problem part.

During the «Elaboration» of the Target Case, the first three functional descriptors (ds1, ds2 and ds3) are absolutely mandatory and no research will be launched if one of them is incomplete.

The ds4 and ds5 descriptors are optional and the absence of their values does not block the discovery but it can lead to «false-positive» results, especially in the case of ds4. However, the existence of information in these descriptors automatically gives them a mandatory aspect to be considered while the discovery.

Formally, the descriptors of this part are not handled in the same way because they belong to two different categories:

1- Informational descriptors (ds1, ds2 and ds3).
2- Conditional descriptors (ds4 and ds5).

Our system assigned to each descriptor an attribute relative to the information presence and is noted $ds_i^{Presence}$. In the first three descriptors, this attribute is equal to 1 and in the ds4 and ds5 descriptors, it can have the value 0 or 1 depending on the presence of information or not.

In addition, as regards the first three descriptors (Informational descriptors), our system assigns a second attribute relative to the value of the descriptor noted ds_i^{Value}. It reflects the considered concepts from the hierarchical model of the used domain ontology.

The first three descriptors of the problem's functional part (ds1, ds2 and ds3) have therefore two attributes related to the presence of the information in the descriptor and the descriptor value:

$ds_i = (ds_i^{Presence}, ds_i^{Value})$.

However, ds4 and ds5 conditional descriptors require a special formalization. Whether it is a precondition or a postcondition, a conditional descriptor expresses a condition that must be met during discovery.

For several years, automatic processing of conditions and constraints has known great success in theoretical and academic point of view as well as practical and industrial one. Thus, several studies have been presented in the literature among which we mention: OCL (Object Constraint Language) [17], TL (Temporal Logic) [18] and FOL (First Order Logic) [19]. These work are distinguished by the power and effectiveness of their calculation in various fields, by using effective tools called constraint solvers.

However, the use of these work requires a mastership in the treatment of constraints and complex predicates, specifically in writing programs and the declaration of the constraints' programming. This constitutes a major obstacle to the use of these work in the expression of constraints.

So, faced to this problem we chose to associate a condition, regardless of its type (precondition or postcondition), to an atomic formula stating a client's constraint. This combination does not only facilitate expressing conditions in a simple format to be handled by users of our system, but also matching descriptors between the client's query and their corresponding in existing services concerned by these conditions.

Our atomic formula is a constraint on a given concept of the used domain ontology. Thus, this

ontological concept is compared to a specific value (instance) via a comparison operator (=,! =, <,>, ≤ or ≥). We recall that each one of the descriptors ds4 and ds5 is a list that may contain one or more elementary conditions. In the context of complex or composite conditions we distinguish the use of operators «Logical AND » and «Logical OR». These are combinatorial logic functions directly issued from mathematical (Boolean algebra), which are the basic tools of programming constraints.

When ds4 or ds5 descriptors have several elementary conditions, they are implicitly linked by « Logical AND », forming as a result a complex condition that requires the validity of all its components.

Regarding the «Logical OR» in the case of preconditions and postconditions of a Web service, the use of this operator has only a real interest within the same elementary condition. It allows the service provider and the customer to express a condition opened to several choices that can be defined among the instances of the operated concept.

We illustrate the use of both logical operators within the descriptor ds4 in the following example. Given a client seeking an online flight booking service. In the description of its query, the client wants to express its requirements in terms of preconditions by the following list:

(PayementMode =«Visa Electron» <u>OR</u> «American Express»)

<u>AND</u>

(Airline.Co = « MEA Airlines »).

Our client expresses two preconditions:

The first one concerns the payment mode; he requires that its payment would be through a « Visa Electron » card or an « American Express » card. It is this ambition that the operator « Logical OR» is used. The second precondition regards the airline company. The client requires only a booking with «MEA Airlines».

However, in the treatment of the descriptor ds4, the operator « Logical AND » is considered between the two preconditions, forming therefore a complex condition requiring the validity of each of its components.

In order to formalize our functional conditions, we use the following triplet to represent an Atomic Formula (AF) such as:

AF= (C, V, O) where:

- C: represents the operated concept. Normally it should be a concept of the application domain ontology (color of the car, etc...).
- V: represents the instance(s) assigned to the concept.
- O: indicates the relational operator (=, ! =, <,>, ≤ or ≥).

We assign four attributes to each one of the two conditional descriptors in the problem functional part (ds4 and ds5). Therefore, they will have:

- $ds_i^{Presence}$ **:** the presence of information in the descriptor,
- $ds_i^{Concept}$ **:** the operated concept,
- ds_i^{Value} **:** the value assigned to the concept.
- $ds_i^{Operator}$ **:** the used operator.

Thus, the descriptors ds4 and ds5 are represented as follows:

$ds_{i=}(ds_i^{Presence}, ds_i^{Concept}, ds_i^{Value}, ds_i^{Operator})$.

After defining the functional descriptors of the customer's query, we return back to our CBR4WSD system, more specifically to the component «Target Case Elaborator ». This component is responsible for completing the description of the Target Case by annotating the community service to which it corresponds. Using functional descriptors of the Target problem, this component must identify from a Community Base, the one which is associated with the Target Case. [20]

The five descriptors presented before are not the sole ones that functionally describe the Case problem part. We supplement this set of functional descriptors by a key descriptor noted ds6. This original descriptor expresses decisive information which allows us to select the search space to be considered in the « Retrieve» phase. Formally, it provides information on the Service Community where belongs the Case.

However, unlike the first five functional descriptors (ds1, ..., ds5) whose values are initiated directly in the client's query, the value assigned to this key descriptor will be deducted after launching the query in our CBR4WSD system. Thus, using the « Goal », fundamental descriptor of the functional part of the client's query, we assign to the descriptor ds6 the identifier of the community which is associated with the Target Case.

After revealing the descriptors of the problem's functional part, we move to the exploitation of the non-functional part details to highlight its descriptors.

B. Non-Functional descriptors of the Case problem part

We recall that the non-functional properties express the conditions when interacting with a given Web service and they are related to different fields.

For example, the endpoint of a service can use messages encrypted by specific cryptographic algorithms. These non-functional properties specify the level of security provided by the Web service when it is accessed thru this endpoint.

Thus, different Web services or even a given Web service using different endpoints may provide the same functional properties with different non-functional aspects. These aspects are, in fact, the essential criteria of the selection process.

The description of the problem's non-functional part is inspired from the non-functional properties of the Web service operation. We have chosen to express them by means of policies (Figure 2).

A policy is a set of policy alternatives, where each alternative is represented by a set of policy assertions. However, in our CBR system, we have chosen to associate a policy in its assertion level to an atomic formula stating a preference of the client. This brings us to represent a policy by one or more atomic formulas. This transformation does not only facilitate the expression of non-functional properties but also the matching between the non-functional properties of the client and the ones of existing services.

The non-functional descriptors of the Case problem part (pb) are represented as ds7 (Table 2).

	Number and name of descriptor	**Definition**
Non-Functional Descriptors (optional)	ds_7 **: <non-functional propretie(s)>**	**List of the desired atomic formulas in the required operation.**

Table 2. Non-Functional descriptors of the Case problem part.

While elaborating the Target Case, the non-functional descriptors are optional and the absence of value at this level does not block the discovery process. This section is specifically used to express preferences of the client and not his requirements.

However, keeping so generic the definition of a non-functional property can lead to matching problems due to a wrong consideration of the constraint semantic concepts. This is why we limit our non-functional properties' diameter to the QoS circle.

Accordingly, our atomic formula represents a constraint on a given characteristic or concept of the employed QoS ontology. Thus, this concept is formally compared to a precise value via a comparison operator (=,! =, <,>, ≤ or ≥).

A client can have multiple non-functional properties. However, these properties don't have the same importance degree in his priorities. He may has some properties much more important than others, hence the need to use a weight assigned to each property so as to indicate its importance to the client.

We recall that the descriptor ds7 is a list that may contain one or more elementary conditions.

In the context of complex or composite conditions we distinguish the use of operators «Logical AND » and «Logical OR». These are combinatorial logic functions directly issued from mathematical (Boolean algebra), which are the basic tools of programming constraints.

When ds7 descriptor has several elementary conditions (assertions), they are implicitly linked by « Logical AND », forming as a result a complex condition that requires the validity of all its components.

Regarding the "Logical OR" in the ds7 descriptor representing Web service non-functional properties, the use of this operator has only a real interest within the same elementary condition. It allows the service provider and the customer to express a condition opened to several choices that can be defined among the instances of the QoS concept in question.

We illustrate the use of both logical operators within the descriptor ds7 in the following example. Given a client seeking an online flight booking service. In the description of its query, the client wants to express its preferences in terms of non-functional properties by the following list:

(Language = « English » OR « French »)
AND
(SecurityEncryption = « RC4 »).

Our client expresses two assertions:

The first one concerns the desired language; he prefers *« English »* or *« French »*. It is this ambition that the operator « Logical OR» is used.

The second preference regards the security. The client prefers a service using the encryption algorithm *«RC4»*.

However, in the treatment of the descriptor ds7, the operator « Logical AND » is considered between the two assertions, forming therefore a complex condition requiring the validity of each of its components.

In order to formalize our non-functional condition, we use the following quadruplet to represent an atomic formula (AF) such as AF= (C, V, O, W) where:

C: Normally it should be a concept of the special ontology of QoS (price, response time, security level, etc..). This does not mean that this parameter cannot be a concept of the application domain ontology (the color of the car, etc..).

- V: represents the instance(s) assigned to the concept. It can be quantitative or qualitative value (number or other).
- O: indicates the relational operator (=, ! =, <,>, ≤ or ≥).
- W represents the weight and it indicates the degree of importance of a non-functional property for a client in his query.

As we have mentioned before, a non-functional property, relative to an operation of a

SAWSDL service, which has been initially expressed in WS-Policy will be represented in the client's query as one or more atomic formulas formalized by the set of attributes «Concept, Value, Operator and Weight ». Thus, in the second part of the problem *(pb)* dealing with non-functional properties, we assign five attributes to each atomic formula of our descriptor. Therefore, each atomic formula will be described by the following attributes:

- $ds_i^{Presence}$: the presence of information in the descriptor.
- $ds_i^{Concept}$: the QoS concept in question.
- ds_i^{Value} : the value assigned to the concept.
- $ds_i^{Operator}$: the used operator.
- ds_i^{Weight} : the weight assigned to the non-functional property.

Thus, the descriptor of the non-functional part is represented as follows:

ds_{7j}= ($ds_{7j}^{Presence}$, $ds_{7j}^{Concept}$, ds_{7j}^{Value}, $ds_{7j}^{Operator}$, ds_{7j}^{Weight})

Table 3 illustrates the overall structure of the problem part of a handling Case in the CBR4WSD system.

Problem Part							
Functional Part							**Non-Functional Part**
ds1	**ds2**	**ds3**	**ds4**	**ds5**	**ds6**		**ds7**
			$ds_{4,1}^{C}$	$ds_{5,1}^{C}$	ds_{6}^{C}	ds_{6}^{V}	$ds_{7,1}^{C}$
			$ds_{4,1}^{O}$	$ds_{5,1}^{O}$			$ds_{7,1}^{V}$
			$ds_{4,1}^{V}$	$ds_{5,1}^{V}$			$ds_{7,1}^{O}$
			…	…			$ds_{7,1}^{W}$
			$ds_{4,n}^{C}$	$ds_{5,m}^{C}$			…
							$ds_{7,n}^{C}$
			$ds_{4,n}^{O}$	$ds_{5,m}^{O}$			$ds_{7,n}^{V}$
							$ds_{7,n}^{O}$
			$ds_{4,n}^{V}$	$ds_{5,m}^{V}$			$ds_{7,n}^{W}$

Table 3. Structure of the problem part of a Case in CBR4WSD.

C. Transformation of a concrete Web service description into a Case

To better understand this migration from the level of Web service standard description to the level of a Case in our CBR4WSD system, we have chosen to illustrate this action with a concrete example. Our example deals with a Flight Management Web service (book a flight, cancel a reservation, confirm a reservation ...).

Fig. 3. Excerpt from the functional description of the "Flight Manager" service

We present in the figure above (Fig. 3) an excerpt from the functional part of our Web service «Flight Manager» which is described in SAWSDL and we focus on the operation «bookFlight» (line 29) for flight booking. This operation is semantically annotated according to our model presented in Fig. 2. We can see clearly the use of the «conceptType» attribute in referring ontological concepts in the «bookFlight» (Figure 3 - line 30). This operation is annotated by by the following concepts: «bookFlight» (goal), «paymentVisa» (precondition) and «sendEmailNotif» (postcondition). In addition to these three new elements that we have incorporated into the SAWSDL description, there are two other descriptive elements in the operation «bookFlight». We are talking about the elements «FlightRequest» (Input) and «FlightResponse» (Output), which in our example are represented by complex structures (lines 4, 17) each consisting of a set of simple elements. As for the non-functional part of our service, we present in Figure 4 an excerpt from the WS-Policy description corresponding to the «bookFlight» operation. In this description, we use special ontologies to express conditions as an atomic formula, specifically the QoS constraints. Thus, an expression composed of quadruplet **<parameter, value, unit, operator>** is associated with each such constraint. This is the case of «ResponseTime» and «ServicePrice» (lines 14, 20). However, the constraints which are not relevant to the QoS are specified in accordance with WS-Policy norms. We define in our example the encoding type «Text Encoding» (line 10) and the language used «Language» (line 11).

```
1  <wsp:Policy
2      xmlns:wsp="..."
3      xmlns:wsse="..."
4      xmlns:qoso=".../ontology/operator"
5      xmlns:qosu=".../ontology/unit"
6      ...>
7
8      <wsp:ExactlyOne>
9          <wsp:All>
10         <wsp:TextEncoding wsp:Usage="wsp:Required" Encoding="Utf-8"/>
11         <wsp:Language wsp:Usage="wsp:Optional" Language="English" wsp:Preference="1"/>
12         <wsp:Assertion name="RTassertion" ...>
13             <wsp:expression>
14                 <wsp:parameter> qosid:ResponseTime </wsp:parameter>
15                 <wsp:Value> 5 </wsp:Value>
16                 <wsp:Unit> qosu:seconds </wspes:Unit>
17                 <wsp:Operator> less </wsp:Operator>
18             </wsp:expression>
19             <wsp:expression>
20                 <wsp:parameter> qosid:ServicePrice</wsp:parameter>
21                 <wsp:Value> 0 </wsp:Value>
22                 <wsp:Unit> qosu:USD </wspes:Unit>
23                 <wsp:Operator> equal </wsp:Operator>
24             </wsp:expression>
25         </wsp:All>
26     </wsp:ExactlyOne>
27 </wsp:Policy>
```

Fig. 4. Excerpt from the non-functional description associated to the «bookFlight» operation

We finalize our example by creating the appropriate Case to the description of the «bookFlight» operation. Table 4 shows the problem part in our Case which reflects the details of the functional and non-functional descriptions of the «bookFlight» operation. Each part has its own descriptors and each descriptor is a set of attributes that will be used later in our calculations of similarities. In our Case example, we use 7 descriptors (ds1, ..., ds7) to cover all the descriptive requirements of the concerned operation We note that in the non-functional part of the Case, the weight attribute assigned to each NFP is not described in the concrete description of the service. Rather it is the client who defines them in its query.

Functional Part				Non-Functional Part		
ds1	Goal	bookFlight		ds7	PNF	$ds^{c}_{7.1}$: TextEncoding
ds2	Input	flightRequest	$ds^{c}_{2.1}$: DepartureCity			$ds^{v}_{7.1}$: Utf8
			$ds^{c}_{2.2}$: ArrivalCity			$ds^{o}_{7.1}$:=
			$ds^{c}_{2.3}$: DepartureDate			$ds^{w}_{7.1}$: 0,5
			$ds^{c}_{2.4}$: AirlineCo			$ds^{c}_{7.2}$: Language
			$ds^{c}_{2.5}$: ClientName			$ds^{v}_{7.2}$: English
			$ds^{c}_{2.6}$: ClientFamily			$ds^{o}_{7.2}$:=
			$ds^{c}_{2.7}$: PassportNb			$ds^{w}_{7.2}$:0,5
ds3	Output	flightResponse	$ds^{c}_{3.1}$: TicketNb			$ds^{c}_{7.3}$: responseTime
			$ds^{c}_{3.2}$: Price			$ds^{v}_{7.3}$: 5 seconds
ds4	Precondition	ds^{c}_{4}: PaymentMode				$ds^{o}_{7.3}$:<
		ds^{v}_{4}: paymentVisa				$ds^{w}_{7.3}$:1
		ds^{o}_{4} :=				$ds^{c}_{7.4}$: ServicePrice
ds5	Postconditio n	ds^{c}_{5}: Notification				$ds^{v}_{7.4}$: 0 USD
		ds^{v}_{5}: sendEmailNotif				$ds^{o}_{7.4}$: =
		ds^{o}_{5} :=				$ds^{w}_{7.4}$: 1
ds6	ServCom	**Deducted descriptor (Value assigned during the processing)**				$ds^{c}_{7.1}$: TextEncoding
						$ds^{v}_{7.1}$: Utf8

Table .4. Problem part of the Case representing the « bookFlight» Operation from the «FlightManager» service**.**

IV. REPRESENTATION OF THE SOLUTION PART OF A CASE HANDLED IN CBR4WSD

In this section, we focus on the formalization of the solution part « *sol(pb)* » of our Case. In our approach, we sought to highlight certain aspects of representation that will help to provide an optimized and efficient solution which is responsive to the problems of non-invocation that may occur. In the following, we detail these aspects that motivated our choices in terms of information needs.

A. Informational needs taken into consideration

When invoking a well-defined operation of a Web service, the client must have information that allows him to precisely locate the requested operation. This is done through information that indicates the service where the requested operation belongs and the service provider contact with whom the client should communicate in case of failure. Also, these properties, alone, are not sufficient to access the requested operation, since a service that implements many operations distributes their physical access thru different endpoints. Each endpoint is the gateway for a physical access to one or more operations. It is therefore imperative to know these technical properties. After selecting the most similar service to the query, the customer may encounter a problem due to non-compliance of its technical properties between its system and the invoked service. This may be due to the communication protocol used to invoke the service or even the data schemas used to represent the service's results. Finally, in the case where many solutions are returned to the client, a further degree related to the criterion of the solution quality is strongly required to guide the choice of the client. He can then select the adequate solution from those responding its functional and non-functional properties. According to the needs outlined above, the information related to the solution part of the Case is divided in three categories:

- Location of the service and its operation. It gives information about the Web service and its URI but also the information concerning the target operation in this Web service.
- Technical characteristics offered by the target operation. It gathers the technical information needed to access the requested operation but also the data schema that models the results provided by the service.
- Client satisfaction degree. This category is described by the percentage of the client satisfaction in terms of functional and non-functional properties and also a degree of an overall satisfaction.

In the following, we use this information to model the solution part of the Case.

B. B. Descriptors of the Case solution part « sol(pb) »

According to the CBR representations, the descriptors of the solution part are noted Ds_i and they are given in Table 5. Finally, we present the solution part « *sol(pb)* » that corresponds to our problem part « *(pb)*» so as to create a full Case.

	Number & Name of the descriptor	Details
Location	Ds_1 : Service	$Ds_{1.1}$: Name
		$Ds_{1.2}$: URI
		$Ds_{1.3}$: Entreprise
		$Ds_{1.4}$: Contact
	Ds_2 : Operation	$Ds_{2.1}$: Name
		$Ds_{2.2}$: URI
Technical Properties	Ds_3 : Access	$Ds_{3.1}$: Communication Protocol
		$Ds_{3.2}$: Port number
		$Ds_{3.3}$: End Point
	Ds_4 : Data Schema	Ds_4 : XSD
Client Satisfaction	Ds_5 : Functional properties	Ds_5 :% of FP satisfaction
	Ds_6 : Non-Functional properties	Ds_6 :% of NFP satisfaction
	Ds_7 : Global	Ds_7 :% of overall satisfaction

Table -5 Descriptors of the solution part of a Case in CBR4WSD

	Number & Name of the descriptor	Details	
Location	Ds_1 : Service	$Ds_{1.1}$: FlightManager	
		$Ds_{1.2}$: "http://localhost:8080"	
		$Ds_{1.3}$: UL	
		$Ds_{1.4}$: Contact	ElBitarCo.
			IbrahimElBitar
			00961xxxxxx
			ibrahim.bitar@ul.edu.lb
	Ds_2 : Opération	$Ds_{2.1}$: flightBook	
		$Ds_{2.2}$: "http://localhost/flightBook.sawsdl"	
Technical Properties	Ds_3 : Accès	$Ds_{3.1}$: http	
		$Ds_{3.2}$: 80	
		$Ds_{3.3}$: "http://localhost/flightBook.sawsdl"	
	Ds_4 : Data Schema	Ds_4 :"http://uddi.org/schema/uddi_v3.xsd"	
Client Satisfaction	Ds_5 : Functional properties	Ds_5 : 80%	
	Ds_6 : Non-Functional properties	Ds_6 : 80%	
	Ds_7 : Global	Ds_7 : 80%	

Table -6 Solution part of the Case corresponding to «bookFlight».

V. TARGET CASE « ELABORATION»

According to Mille [21], the «Elaboration» phase consists in building the Target Case based on the Inputs of the CBR system. This phase has two main tasks: the creation and the preparation of Target Case. To better understand it, we present in the section below how we apply these two tasks within our CBR4WSD system.

A. Creating and preparing a Target Case

During the «Elaboration», we create the new Case according to the description specified to the application domain. However, Fuchs [22] proposed to complete, if possible, a problem description by collecting other relevant information to find the solution of the target problem.

The essential idea was implemented in our CBR4WSD system particularly in the component « Target Case Elaborator ». This component is responsible for completing the description of the

Target Case by annotating the Service Community to which it corresponds. Hence, using the « Goal », fundamental descriptor of the functional part of the client's query, this component identifies the identifier of the Service Community which is associated with the Target Case.

Thus, we completed our set of functional descriptors (ds1, ..., ds5) by a key descriptor noted ds6. This original descriptor expresses decisive information that allows us to select the search space to be considered in the « Retrieve» phase. Formally, it provides information on the Service Community where belongs the Case. As we mentioned before, the value assigned to this descriptor is inferred from the client's query. The second task of the «Elaboration» phase is responsible for preparing the Case, by choosing the appropriate indices for Case research [23]. These indices are some descriptors that are relevant to the resolution of the problem and that constitute the basis of research in the Case Base. This is exactly what we did by determining which descriptors among our Case structure are indispensable to process each new Target Case. Thus, we retained mandatory descriptors such as « Goal, Input and Output» and no discovery process can be launched without these descriptors, unlike other descriptors that are second class matter. This does not undervalue the consistency of the information provided by these descriptors if it exists. This «Elaboration» phase which consists of formalizing the problem description is not as simple. It may be accompanied by problems related to the knowledge handled in Cases. We expose this idea in the next section.

B. Knowledge Imperfection Problems

The CBR is a problem solving paradigm that is based on the reuse of past experiences, stored in a Case Base, to solve new problems called Target Cases. Generally, using experiences or problems, means concrete Cases involving approximate, blurred or vague knowledge, expressed in human language. These knowledge imperfections are due to reasons related to obtaining knowledge by observation and the representation of this knowledge. They range among three different types [24]

- The imprecision concerns digital knowledge as in the case of measurement errors (weight with 1% margin for example) or flexible knowledge (load capacity of a lift up to 4 or 5 persons).
- The uncertainty reflects a doubt about the knowledge validity. It can, for example, be due to a relative reliability of the means used during the knowledge capture. It can also occurs when the observer intentionally give erroneous, incorrect or inaccurate information. Finally, a difficulty in knowledge obtaining or verification and forecasts are also examples of cases that lead to uncertainties.
- As for the incompleteness, it constitutes a total or partial absence of knowledge. It is, in general, due to the inability to obtain certain information (e.g. forms which are not completely filled.) or to a problem occurring at the time of the knowledge capture (e.g. image with a hidden part).

After presenting the problems generally faced in CBR systems, we return to our own system to present the situation. In fact, in our system specifically designed for Web services discovery, we do not find all problems of knowledge imperfection.

The Target problem is structured in such a way as to describe in detail the properties of the sought operation of a service. However, the descriptors of this part of the Case do not include data that can withstand uncertainties or inaccuracies. The handled Target problem does not reflect a real perception as in the case of medical and industrial diagnosis, where we can easily fall on imprecise and uncertain data. In our system, it is rather the solution we seek that is real and concrete. Thus, the only problem of imperfect knowledge that we encounter in our Case is the incompleteness since the definition of our descriptors admits both mandatory and optional descriptors. No Target Case will be generated if the client tries to run a query with incomplete data at mandatory descriptors (Goal, Inputs and outputs). However, if it is the case in optional data, we treat the problem of incompleteness in the «Retrieve» phase without having to modify or complete incomplete descriptors, due to the unpredictability of the preconditions or postconditions of a client or even its non-functional properties.

VI. CONCLUSION

In this paper, we have focused on the « Elaboration » phase of a Case in our CBR4WSD system which is dedicated for the automatic discovery of Web services. We have described how to represent our Web service Case, while respecting our departure goal as regards the alignment with W3C standards. Thus, based on our enriched semantic Web service description model, we have extracted our needs in terms of data or significant information to formalize our Case in its two parts « pb » and « sol(pb) ». A general structure of a Case handled in our CBR4WSD system was also presented. In the study of the « Elaboration » phase, we have also exposed the tasks of creating and preparing Cases and we have presented their application in the specific context of our system. Finally, we have presented the problems of

knowledge imperfections that may be encountered in CBR systems and more specifically in our CBR4WSD system.

REFERENCES

[1] Gebhardt, F.,Vob, A.,Grather, W. et Schmidt-Beltz, B. Reasoning with Complex Cases, Kluwer Academic, Norwell, MA.(1997)

[2] Omrana H., El Bitar I., Belouadha F-Z., Roudies O.: A Comparative Evaluation of Web Services Description Approaches, 10th International Conference on Information Technology : New Generations ITNG 2013 April 15-17, 2013, Las Vegas, Nevada, USA.

[3] Klusch.M., CASCOM Intelligent Service Coordination in the Semantic Web. (2008). Volume: 16, Issue: 2, Publisher: Birkhäuser Basel, Pages: 31-57

[4] Martin D. et al., OWL-S: Semantic Markup for Web Services. W3C Submission, 2004. http://www.w3.org/Submission/OWL-S/.

[5] Bruijn J. et al., Web Service Modeling Ontology (WSMO). W3C Member Submission, 2005. http://www.w3.org/Submission/WSMO/.

[6] Christensen E., Curbera F., Meredith G., and Weerawarana S., Web Services Description Language (WSDL) 1.1. http://www.w3.org/TR/wsdl, March 2001

[7] Chung L., Nixon B.A., Yu E. and Mylopoulos J., "Non-Functional Requirements in Software Engineering", KluwerAcademic Publishers, Boston Hardbound, October 1999.

[8] Nadalin, A., M. Goodner, M. Gudgin, A. Barbir, and H. Granqvist (2009). WS-SecurityPolicy 1.3. OASIS standard, http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/ws-securitypolicy.pdf.

[9] WS-Reliability 1.1. OASIS Standard, 2004. Available at http://docs.oasis-pen.org/wsrm/ws-reliability/v1.1/wsrmws_reliability-1.1-spec-os.pdf.

[10] Osman T., Thakker D., Al-Dabass D.; Semantic-Driven Matchmaking of Web services using Case-Based Reasoning. In the fourth IEEE International Conference on Web Services (ICWS 2006) (pp. 29-36). Chicago, USA.

[11] Osman T., Thakker D., Al-Dabass D.; S-CBR: Semantic Case Based Reasoner for Web services discovery and matchmaking. In 20th European Conference on Modeling and Simulation (ECMS 2006) (pp. 723-729). Bonn, Germany.

[12] Lajmi S., Ghedira C., Ghedira K., « How to apply CBR method in web service composition », 2nd International Conference on Signal-Image Technology & Internet based Systems (SITIS'2006), Springer Verlag ed. Hammamet (Tunisie). LNCS series, 2006

[13] Lajmi S., Ghedira C., Ghedira K., Benslimane D., « Wesco_cbr : How to compose web services via case based reasoning », IEEE International Symposium on Service-Oriented Applications, Integration and Collaboration held with the IEEE International Conference on e-Business Engineering (ICEBE 2006), Shanghai, China, 2006

[14] Wang L., and Cao J.; Web Services Semantic Searching enhanced by Case Reasoning, 18th International Workshop on Database and Expert Systems Applications, 2007.

[15] De Franco Rosa F. et De Oliviera J.M. ; An approach to search Web Services using Ontologies and CBR, The 11th IEEE International Conference on Computational Science and Engineering – Workshops, 2008.

[16] El Bitar I., Belouadha F.Z, Roudies O., "Towards a semantic description model aligned with W3C standards for WS automatic discovery". The 4th International Conference on Multimedia Computing and Systems (ICMCS'14) April 14-16 2014, Marrakesh, Morocco

[17] Warmer J., Kleppe A., The Object Constraint Language: Getting Your Models Ready for MDA, book, Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA ©2003 ISBN:0321179366.

[18] Benthem V. and Fak J., Tense Logic and Standard Logic in Tense Logic. Logique et Analyse Louvain, 1977, vol. 20, no 80, p. 395-437.

[19] Codognet P., Programmation logique avec contraintes : une introduction Technique et Sciences Informatiques, 1995 - info.ucl.ac.be

[20] El Bitar I., Belouadha F.Z, Roudies O., "A CBR based approach for web service automatic discovery". Journal of Theoretical and Applied Information Technology (ISSN 1992-8645), Volume 62 No 1 2014, http://www.jatit.org

[21] Mille, A. "Tutorial: raisonner à partir de cas: principe, théorisation et ingénierie de

la connaissance associée" ; présenté en 14e Atelier du Raisonnement à Partir de Cas, Besançon, France.(2006)

[22] Fuchs B. , Représentation des connaissances pour le raisonnement à partir de cas, le système ROCADE, Thèse de doctorat, Université Jean Monnet de Saint-Etienne, France.(1997)

[23] Fuchs B., Lieber, J., Mille, A., et Napoli, A. " Une première formalisation de la phase d'élaboration du raisonnement à partir de cas", le 14e Atelier de Raisonnement à Partir de Cas, Mars 2006, Besançon, France.

[24] El Bitar I., Belouadha F.Z, Roudies O., "A Logic and Adaptive Approach for Efficient Diagnosis Systems using CBR", International Journal of Computer Applications 39(15):1-5, February 2012. Published by Foundation of Computer Science, New York, USA. BibTeX (ISSN 0975 – 8887)