Case-based Matching Algorithm for Dynamic Web Service Discovery

Ibrahim El Bitar
Department of Applied Mathematics & Computer Science, Faculty of Sciences, Lebanese University, Lebanon.

Zouhair Bazzal
Departement of Computer and Communication Engineering, School of Engineering, Lebanese International University, Lebanon.

Fatima-Zahra Belouadha
SIweb Team, SIR Laboratory Ecole Mohammadia d'Ingénieurs, Mohammed Vth University-Agdal, Rabat, Morocco

ABSTRACT

With the increasing number of Web services available on the web, looking for a particular service has become very difficult, especially with the evolution of the clients' needs. In this context, we have previously proposed the CBR-based system for semantic Web service discovery (CBR4WSD) which benefits from the advantages of CBR to address the limitations of existing approaches in terms of efficiency of the Web service selection. This paper is devoted to the study of the Retrieval phase, which is the core of our CBR4WSD system. First, we expose the stage of Retrieval preparation which is performed in the Offline discovery process. Then, we present the discoverability checking-rules that help our system to detect the feasibility of the discovery process for a given query. We also present our Retrieval algorithm that calculates the functional and the non-functional similarities before generating the global similarity measure.

Keywords

Semantic Web Service Discovery, Matchmaking, CBR, Formalization, Retrieval, Functional properties, Non-Functional properties

1. INTRODUCTION

The semantic Web service (WS) discovery has been given massive attention within the last few years. With the increasing number of WS available on the web, identifying a particular service has become very difficult, especially with the evolution of the requesters needs, those who have become more and more demanding. In this context, various approaches to discover semantic WS have been proposed. The integration of semantics in the WS description has undoubtedly improved their interpretation and subsequently, their discovery process by identifying and selecting the appropriate services. However, the integration of semantics does not mean the automation of the discovery process, especially with the need for human intervention to refine the results in many existing approaches [1].

The enduring need for discovery automation has involved Artificial Intelligence reasoning to guide a dynamic WS discovery. In the range of the intelligent works, the Case Based Reasoning (CBR) has scored a great success in the field compared to existing works as it offers the opportunity of reusing successful old experiences to solve new problems, specifically in the case of WS, where normally the behavior of a service is difficult to presume before its execution. However, the existing CBR-based approaches for WS discovery present some limitations that researchers are trying to fulfill [2][3][4][5][6][7][8]. In fact, they present problems concerning Case representation and expressiveness, semantic annotation and ontology use, Case retrieval and matchmaking process and finally Case Base organization and indexing.

Thus, we have previously proposed our CBR4WSD (CBR for Web Service Discovery) approach. It is a CBR-based approach for semantic WS discovery. Its outline contribution includes a set of aspects which aim to overcome the limitations of existing approaches and gives the originality of our CBR4WSD approach. These aspects are mainly related to the processing rationalization, the control and mastery of the treated WS volumetry and the alignment with standards, without forgetting the improvement of the results' quality in terms of their ability to meet both functional and non-functional clients' needs [9].

The remainder of this paper is structured as follow. The second section describes the general process of our CBR4WSD approach. Section 3 highlights the formalization of the Cases handled in CBR4WSD. The fourth section firstly provides details about the semantic matching process between ontology concepts and explains the need of performing this calculation in the Offline discovery process. Then, it focuses on the Case Retrieval phase where we present our algorithm of semantic matchmaking in its three stages: Functional Similarity, Non-Functional Similarity and Global Similarity. The fifth and final section releases a conclusion of the presented work.

2. CBR4WSD PROCESS

In this section, we expose the WS discovery process in our CBR4WSD approach. As shown in Fig. 1, this process starts with the transformation of the client's query into a Target Case aligned with the W3C standards: SAWSDL and WS-Policy. This operation is performed by a semantic Target Case generator and it consists in representing the client's query as a Target Case, where its problem part is described by a set of semantic descriptors reflecting the functional and non-functional properties. The solution part descriptors are kept unspecified at this phase and the discovery process will attempt to instantiate them with a discovered WS. Alternatively, the problem descriptors represent an abstract WS, while the solution descriptors represent the corresponding concrete WS.

The generated Target Case passes into the Target Case Elaborator that is responsible for completing the Target Case description by annotating the service community to which it corresponds. It uses the functional descriptors to identify the corresponding Service Community from the Community Base.

Moreover, the overall discovery process continues by retrieving Source Cases that are similar to the Target Case. At this level, after identifying the Source Cases that belongs to the same service community as the Target Case, the "Semantic Online Matchmaker" proceeds to the calculation of their functional similarity measure (FSM) alongside the Target Case. Upon completion of this operation, a set of Source Cases whose calculated FSM meets a defined threshold will be retrieved. The set of retrieved Source Cases is then projected into a selector that identifies the best Cases fulfilling the required non-functional properties. This component performs a "matching" applied on the Source and Target policies to calculate the non-functional similarity measure (NFSM) and subsequently generate the global similarity measure (GSM) between the Source Cases and the Target Case. The goal is to identify the WS that will be recommended to the client. These WS are the solutions associated with the Source Cases whose GSM are the highest.

Finally, after receiving the results of the test performed by the client, only the satisfying WS are used to instantiate the solution descriptors of the Target Case. Resulting Cases will be introduced in the Case Base via the component called "Case Retainer".

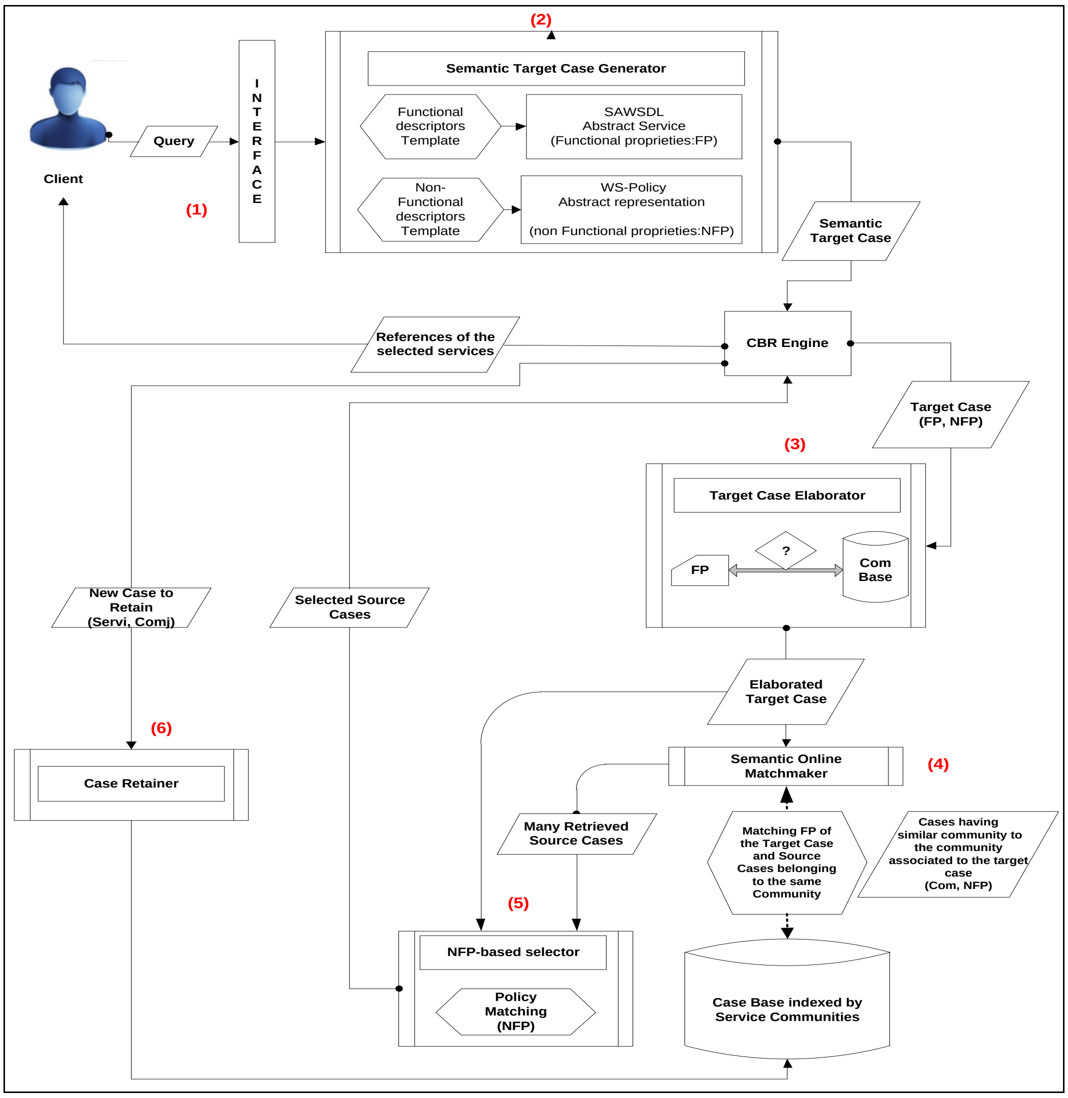
(1)
(2)
(3)
(4)
(5)
(6)
Client
Query
INTERFACE
Semantic Target Case Generator
Functional descriptors Template
SAWSDL Abstract Service (Functional proprieties:FP)
Non-Functional descriptors Template
WS-Policy Abstract representation (non Functional proprieties:NFP)
Semantic Target Case
CBR Engine
References of the selected services
Target Case (FP, NFP)
Target Case Elaborator
FP
?
Com Base
New Case to Retain (Servi, Comj)
Selected Source Cases
Elaborated Target Case
Semantic Online Matchmaker
Case Retainer
Many Retrieved Source Cases
Matching FP of the Target Case and Source Cases belonging to the same Community
Cases having similar community to the community associated to the target case (Com, NFP)
NFP-based selector
Policy Matching (NFP)
Case Base indexed by Service Communities

Fig 1: WS Discovery process in the CBR4WSD approach

3. REPRESENTATION OF THE CASE PROBLEM PART

The Case problem part « (pb) » reflects the client's query seeking a specific operation of a particular service. Our representation of the problem part of a Case handled in CBR4WSD is based on our definition of the WS discovery mechanism as "the act of locating a machine treatable description, of a previously unknown WS describing some functional and non-functional requirements". We consider these functional and non-functional requirements in the Case formalization that stands on our enriched WS description model [10]. Thus, in the Case problem part, we distinguish the functional properties (FP) from the non-functional properties (NFP), hence the notation:

pb = (FP, NFP)

3.1 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.

Table 1. Functional descriptors of the Case problem part.

		Number and name of descriptor	**Definition**
Functional Descriptors	**Mandatory Descriptors**	**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.
	Optional Descriptors	**ds4 : <Preconditions>**	List of preconditions imposed on the required operation.
		ds5 : <Postconditions>	List of postconditions imposed on the required operation.
	Deducted Descriptor	**ds6 : CommunityID**	Service Community where the Case belongs

During the «Elaboration» of the Target Case, the first three functional descriptors (ds1, ds2 and ds3) are absolutely mandatory and no discovery 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.

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 adopted 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:

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

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. Based on the study we have done on the conditions and constraint language representation [11][12][13], 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 domain ontology. Thus, this ontological concept is compared to a specific value (instance) via a comparison operator (=, ! =, <,>, ≤ or ≥).

In order to formalize our functional conditions, we use the following 5-tuple to represent an atomic formula (AF) such as: AF= (C, V, O, U, W) 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 ≥).
- U: represents the unit whether the concept is measurable (quantitative variable).
- W: represents the weight, by default it is equal to 1 in ds4 and ds5.

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

- $\mathrm{ds_i^{Presence}}$: the presence of information in the descriptor,
- $\mathrm{ds_i^{Concept}}$: the operated concept,
- $\mathrm{ds_i^{Value}}$: the value assigned to the concept.
- $\mathrm{ds_i^{Operator}}$: the considered operator.
- $\mathrm{ds_i^{Unit}}$: the concept unit.
- $\mathrm{ds_i^{weight}}$: the descriptor's weight (by default it is equal to 1).

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

$$\mathbf{ds_i} = (\,\mathbf{ds_i^{Presence}},\ \mathbf{ds_i^{Concept}},\ \mathbf{ds_i^{Value}},\ \mathbf{ds_i^{Operator}},\ \mathbf{ds_i^{Unit}},\ \mathbf{ds_i^{Weight}}).$$

We recall that each one of the descriptors ds4 and ds5 is a list that may contain one or more elementary conditions. We consider the use of the operators «Logical AND » and «Logical OR» in the context of complex or composite conditions.

After defining the functional descriptors of the client'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 service community to which it corresponds. Using the problem functional descriptors, the «Target Case Elaborator» must identify from the Community Base, the one which is associated with the Target Case [9].

The five descriptors presented before are not the sole ones that functionally describe the Case problem part. In fact, according to Fuchs [14] who proposed to complete, if possible, a problem description by collecting other relevant information to find the solution of the Target problem, we supplement the 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 Target 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.

3.2 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. We have chosen to express them by means of WS-Policy (W3C recommendation) [10] [15].

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 through this endpoint.

Thus, a Web service using different endpoints can 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 expressed as `policies`. A policy is a set of `policy alternatives`, each represented by a set of `policy assertions`. However, in our CBR system, we have chosen to represent a `policy` by a `policy alternative` consisting of one or more atomic formulas. Each atomic formula represents an assertion that corresponds to a certain preference of the client. This choice aims not only to facilitate the expression of non-functional properties, but also the descriptors' matching between the client's non-functional properties and the ones in the existing services.

W3C Office introduced the concept of "Policy Subject" to finely associate policies to the WSDL elements. To do this, it defines four types of Policy subjects: Service Policy subject, Endpoint Policy subject, Operation Policy subject and Message Policy subject [16]. However, we distinguish three types of exchanged messages: InMessage (Message for an input message) OutMessage (Message for an output message) and FaultMessage (Message for an error message input / output). To cover all of these subjects, we define seven non-functional descriptors in part of a case, as shown in Table 2.

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 considered QoS ontology. Thus, this concept is formally compared to a precise value via a comparison operator (=,! =, <,>, ≤ ,≥, Applied to,…).

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

	Number and name of descriptor	**Definition**
Non-Functional Descriptors	**ds7 : <ServicePolicy>**	List of policy assertions desired on the service.
	ds8 : <OperationPolicy>	List of policy assertions desired on the operation.
	ds9 : <EndpointPolicy>	List of policy assertions desired on the endpoint.
	ds10 : <InMessagePolicy>	List of policy assertions desired on the input message.
	ds11 : <OutMessagePolicy>	List of policy assertions desired on the output message.
	ds12 : <FaultMessagePolicy>	List of policy assertions desired on the fault message.
	ds13 : <BondingPolicy>	List of policy assertions desired on the binding.

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 each one of the Non-Functional descriptors is a list that may contain one or more elementary conditions. We consider the use of the operators «Logical AND » and «Logical OR» in the context of complex or composite conditions.

In order to formalize our non-functional condition, we use the following 5-tuple to represent an atomic formula (AF) such as:

AF= (C, V, O, U, 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 (=, ! =, <,>, ≤ , ≥, applied to).
- U: represents the unit whether the concept is measurable (quantitative variable).

- 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^{Unit} : the concept unit.
- ds_i^{Weight} : the weight assigned to the non-functional property.

Thus, the descriptors of the non-functional part are represented as follows:

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

4. SEMANTIC MATCHING MECHANISM AND OFFLINE DISCOVERY PROCESS

The semantic matching process must take into account possible similarities between the concepts used in the definition of each Case descriptor. Ontology is used to semantically describe the concepts of a given domain and their different properties. These concepts are linked together through semantic relationships providing a hierarchical ontology structure. Degrees of semantic matching between concepts belonging to an ontology tree are generally classified in the literature into four categories [17], namely, "Exact", "Plug-in", "subsumed" and "Fail" degrees.

- « Exact » Similarity:

  The similarity between two concepts C1 and C2, denoted Sim (C1, C2), is considered "Exact" if both C1 and C2 concepts are equivalent, i.e. they belong to the same ontological class (annotated by same URI in an ontology). To express the correspondence « Exact » we adopt the notation: $C1 \equiv C2$.

- « Plug-in » Similarity:

  The similarity between two concepts C1 and C2 is considered a « Plug-in» match if C1 is a subclass of C2, i.e. if the URI used to annotate the concept C1, references in the considered ontology, a concept defined as a subclass of the concept referenced by the URI used to annotate the concept C2. To express the "Plug-in" matching between C1 and C2 we adopt the notation: $C1 \subset C2$.

- « Subsumes » Similarity:

  The similarity between two concepts C1 and C2 is considered a « Subsumes » match if C1 is a superclass of C2, i.e. if the URI used to annotate the concept C1, references in the considered ontology, a concept defined as a superclass of the concept referenced by the URI used to annotate the concept C2. To express the « Subsumes » matching between C1 and C2 we adopt the notation: $C1 \supset C2$.

- « Fail » Similarity:

  The similarity between two concepts C1 and C2 is considered a « Fail » match if they have no relationship or equivalence relationship semantics, i.e. they are annotated by URI referencing two concepts with no equivalence link or semantic relationship in the ontology where they are defined.

  To express the « Fail » correspondence between concepts C1 and C2, we adopt the notation: $C1 \neq C2$.

According to our overall discovery procedure which consists of an Offline and an Online processes, the Offline process is used for configuration and system administration.

During the offline process, we apply the similarity rules between the concepts of the domain ontology, and we generate a table of similarity denoted *FuncSimTab*. Similarly, we apply these rules of similarity between concepts of Non-Functional ontology to generate a second table of similarity denoted *NonFuncSimTab*. Furthermore, the similarity calculation is a time consuming process. The choice of performing the inter-concepts similarity calculation in Offline mode is justified by the purpose of optimizing the Online discovery process. In particular, it should significantly reduce the time allotted for matching Source Cases to the Target Case. Besides, this practice allows avoiding duplicating the calculation of semantic similarity between a concept of the Target Case and another concept that appears in several Source Cases. Therefore, we generate our semantic similarity tables only once regardless of the clients' queries and also the Source Cases treated for each query.

5. CBR4WSD LAYERS AND MODELS

Once the client launches his query, before starting the retrieval algorithm, we apply the discoverability verification rules on the new Target Case. These rules are strict and their results inform the system whether the discovery process is feasible or not in its Case Base.

5.1 Discoverability Verification Rules

Rule 1: Check if the WS Community corresponding to the Target Case is not empty.

The Target Case generator is responsible to complete the description of the Target Case by annotating the community service to which it corresponds. However, using the function *Verify_Community( )* we check if the community is not empty before starting the matching algorithm. This function takes the elaborated Target Case as input parameter and returns (True) if the relevant community is not empty. Otherwise, a notification is sent to the administrator and the discovery process ends with the fact of return (False), certainly after displaying a failure message to the client.

Rule 2: Check the discovery feasibility in the selected community.

In case that Rule 1 returns "True" the system continues processing the discoverability verification within the community corresponding to the Target Case, checking a single condition concerning Outputs. Indeed, the community must be able to satisfy all Outputs of the client's query. Thus, each Output concept in the Target Case must necessarily find its corresponding concept having “Exact” match or “Plugin” match in the Output list of the community LC(O). There is no constraint on the Inputs in this first step. Inputs of the Target Case can find their corresponding or not in the list of Input concepts of the community LC(I). What matters are the Outputs or the outcomes.

We use the function *VerifyOutpTotalMatch( )* to check if all the outputs of the query have their “Exact” or “Plugin” correspondents from the LC (O). This function takes as input parameters the elaborated Target Case and the Community where it is associated and returns (True) if all the query outputs have their Exact or Plugin correspondents in the list LC(O) of the community.

Rule 3: Create the set E and verify that it is not empty

If *VerifyOutpTotalMatch( )* function is verified, the system can initiate the discovery process in the selected community and continues processing through the creation of the set E consisting of the Community Source Cases satisfying the following conditions:
In terms of Outputs, the Source Case must meet at least the Outputs of the Target Case. We can’t manipulate a Source Case that does not guarantee all client’s Outputs. The Source Case Outputs must all have their « Exact » or « Subsumes » correspondents in the Outputs of the query or the Target Case, otherwise the Case will be dismissed.

In terms of Inputs, the Source Case should not require more than the client. We cannot ask the client for Inputs that do not belong to its context. The Source Cases Inputs must have their « Exact » or « Plugin » correspondents in the inputs of the query otherwise the Case will be dismissed.

Moreover, we can come across a Source Case that in terms of Outputs provides much more than the Outputs requested by the client and in terms of Inputs the Target Case satisfies the Inputs of the Source Case and maybe more. This will not cause any problem in the calculation of matching because in both cases, the needs are necessarily satisfied.

After creating the set E comprising Source Cases that verify the client’s needs (Inputs and Outputs) we proceed to calculate the functional similarity of each of its Source Cases with the Target Case.

5.2 Functional similarity algorithm

To calculate the functional similarity between a $\mathrm{Case_i}$ belonging to set E and the Target Case, we proceed by the aggregation of local similarities performed on each of the elements of the problem functional part of a Case illustrated by the following set (Goal, Inputs, Outputs, Preconditions, Postconditions and ComId). However, the computation of local similarity on ds1 and ds6 descriptors is already assumed since we are handling the Source Cases from the set E which necessarily have the same Goal as the Target Case and of course belonging to the same Service Community.

Though, the computation of local similarity between the descriptors ds2, ds3, ds4 and ds5 is complex, since they are expandable lists of variable size. We need a method to measure the degree of similarity between multi-valued descriptors having each one several attributes that are not necessarily ordered.

To do this, we must first choose a similarity measure that complies with our application domain. Generally, the calculation of the local similarity depends on the type of descriptor and is based on the distance. The literature refers to several similarity measures of this type. However, none of these measures is standard. Therefore the choice depends on the field of application concerned. For the functional similarity between the WS Cases, we choose the Manhattan distance having the following formula:

$$\mathrm{Sim}=\frac{\sum_{i=1}^{n}[\mathrm{sim_i}(a_i\ ,b_i)]}{n} \qquad (1)$$

Where: n is the number of attributes and $\mathrm{sim_i}$ is the local similarity calculated for the attribute i.

We use the function *FunctionalSimilarity( )* that compares the functional parts of a Source Case and Target Case given as input parameters, while browsing their descriptors and their attributes to calculate their local similarities. These local similarities are aggregated by means of the Manhattan formula to generate the degree of functional similarity between the Source Case and the Target Case.

We apply the function *FunctionalSimilarity( )* on each Case of the set E to generate its functional similarity against the Target Case.

After performing our functional treatment, only Cases, having a degree of similarity higher than the threshold of Functional Similarity (FSThreshold), will be considered in the next step where a measure of Non-Functional Similarity (NFSM) is calculated. The similarity threshold is defined by the domain expert and depends on the application domain. In the case of WS, two services are functionally similar if their functional similarity degree is greater than or equal to 60%. Thus, two Service Cases are not similar if the differences between their attributes are obvious i.e. if the degree of similarity does not exceed the threshold defined by the domain expert.

5.3 Non-Functional similarity algorithm

We apply the function NonFunctionalSimilarity on the Cases of the set E that have a functional similarity degree (FSimDeg) greater than or equal to 60% to generate their Non-Functional Similarity against the Target Case.

This function takes as input parameters a Source Case from the selected ones and the Target Case (query), and returns the degree of Non-Functional similarity by matching the Non-Functional descriptors of the two compared Cases.

5.4 Global Similarity

After performing our non-functional treatment, only Cases, having a degree of similarity higher than the threshold of Non-Functional Similarity (NFSThreshold), will be considered at the final stage where a measure of overall similarity (GSM) is calculated. The non-functional similarity threshold is defined by the domain expert and depends on the application domain. In the case of WS, two services are non-functionally similar if their Non-Functional Similarity degree is greater than or equal to 60%. We complete our algorithm by the final treatment that allows first to select among the set E, the Source Cases meeting the clients’ Functional and Non-Functional requirements and then calculate their global similarity measure (GSM) against the Target Case.

6. SYNTHESIS

In this paper, we have focused on the « Case Retrieval » phase in our CBR4WSD system which is dedicated for the automatic discovery of WS.

We have firstly presented an overall view of our CBR4WSD system. Then we have described how to represent our WS

Cases, while respecting our initial goal as regards the alignment with W3C standards. Thus, based on our enriched semantic WS description model, we have extracted our needs in terms of data or significant information to formalize our Case « pb » part.

In the study of the « Case Retrieval » phase, we have exposed both Offline and Online discovery process. In the Offline process we have exposed the importance of calculating the similarity between the ontology concepts before launching the Online process so as to reduce the discovery process time.

In addition, it was necessary to define a mechanism of "Case Retrieval" focused on the specific needs of each client, ensuring efficient and rapid selection of WS. To do this, we have proposed a semantic matching algorithm to calculate the functional similarity measure (FSM) consisting of the aggregation of local similarities performed on Cases functional descriptors. Only Cases having a functional similarity degree greater than or equal to the functional similarity threshold are considered in the next step where a measure of Non-Functional Similarity (NFSM) is calculated. This measure is the aggregation of local similarities performed on the Cases non-functional descriptors. This latter is self-aggregated to the (FSM) to generate a global similarity measure (GSM) that identifies the most appropriate services which better meet both functional and non-functional requirements expressed in the Target Case.

Finally, our approach allows, through its enriched semantics and its coverage of functional and nonfunctional aspects, to not only hearten the discovery of the WS responding to the client's query but also to select the finest from the top covering services of non-functional properties using its efficient Retrieval algorithm. This algorithm spreads over CBR mechanisms and the indexation of the Case Base by service communities in order to rationalize the processing and to optimize the time of discovery and the performance of the Retrieval phase.

7. REFERENCES

[1] 1. El Bitar, I., Belaoudha, F.Z., Roudies, O.: Semantic Web Service Discovery Approaches: Overview and Limitations. International Journal of Computer Science and Engineering Survey, ISSN: 0976-2760, Volume 5, Number 4, 2014.

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

[3] S. Lajmi, C. Ghedira, K. Ghedira, "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, 2006a.

[4] S. Lajmi, C. Ghedira, K. Ghedira, D. Benslimane, "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, 2006b.

[5] T. Osman, D. Thakker, D. Al-Dabass; "Semantic-Driven Matchmaking of Web Services Using Case-Based Reasoning". School of Computing and Informatics, Nottingham Trent University, Nottingham, UK, Naval Academy of France, Ecole Navale, BP 600, 29240 Brest, France, 2006

[6] T. Osman, D. Thakker, D. Al-Dabass; "S-CBR: Semantic Case Based Reasoner for Web Services Discovery and Matchmaking", School of Computing and Informatics, Nottingham Trent University, Nottingham, UK, Naval Academy of France, Ecole Navale, BP 600, 29240 Brest, France, 2006.

[7] Z. Sun, J. Han, D. Ma. "A Unified CBR Approach for Web Services Discovery, Composition and Recommendation", 2009 International Conference on Machine Learning and Computing IPCSIT vol.3 (2011) © (2011) IACSIT Press, Singapore.

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

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

[10] El Bitar, I., Belaoudha, 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.

[11] 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.

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

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

[14] 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)

[15] Web Services Policy 1.2 - Framework (WS-Policy) : http://www.w3.org/Submission/WS-Policy/

[16] Web Services Policy 1.5 – Attachment W3C Recommendation 04 September 2007: http://www.w3.org/TR/ws-policy-attach/#ServicePolicySubject

[17] Hatzi, O., Vrakas, D., Bassiliades, N., Anagnostopoulos, D., and Vlahavas, I. (2011). The PORSCE II frame-work: Using AI planning for automated semantic web service composition. The Knowledge Engineering ReView, 2011.