S-CBR: SEMANTIC CASE BASED REASONER FOR WEB SERVICES DISCOVERY AND MATCHMAKING

Dhavalkumar Thakker, Taha Osman, David Al-Dabass
School of Computing & Informatics, Clifton Lane, Nottingham Trent University,
Nottingham, NG11 8NS, United Kingdom
{Dhavalkumar. Thakker, Taha. Osman, David.Al-Dabass}@ntu.ac.uk

ABSTRACT

Web services are being increasingly adopted as the computing engine of choice for today's Internet-driven applications. The composition of Web services further advances their advantages by allowing different services to integrate for providing new, value-added services. The automated discovery of adequate Web services is the pre-requisite and core feature for achieving dynamic Web services composition. In this paper, we present a novel approach that utilizes Case Based Reasoning methodology for modelling the Web services discovery and matchmaking problem. Our framework uses OWL semantic descriptions extensively for implementing both the components of the CBR engine and the matchmaking profile of the Web services.

1. INTRODUCTION

The last decade has witnessed an explosion of application services delivered electronically, ranging from e-commerce and Internet information service, to services that facilitate trading between business partners, better known as B2B relationships. Traditionally these services are facilitated by distributed technologies such as RPC, CORBA, RMI, and more recently Web services.

Web services use XML extensively for messaging, discovery, and description. The use of standardized XML-based messaging makes Web services platform and language neutral. The Web services technology uses Simple Object Access Protocol for XML messaging, which in turn uses ubiquitous HTTP for the transport mechanism. As firewall bypasses HTTP, it allows the Web services to be exposed beyond the firewall. The above advantages make Web service the best-suited computing engine for today's Internet-driven applications. Moreover, the composition of Web services adds a new dimension to Web services advantages by providing value-added integrated service to the end-user or facilitating co-operation between business partners.

Automation prospects related to Web services composition, discovery and matchmaking are being explored to make Web services ubiquitous with open plug-in interfaces. Automatic Web service discovery and matchmaking is the principal aspect for the dynamic service composition. The accuracy of the matchmaking process enhances the possibility of successful composition, eventually satisfying the user and application requirements. The current standard for Web service discovery UDDI is syntactical and has no scope for automatic discovery of Web services. Hence, current approaches attempting to automate discovery and matchmaking process apply semantics to the service descriptions. These semantics are interpretable by the service agents and should include WSDL-based functional (Dean et al. 2005) parameters such as the Web services input-outputs and non-functional parameters (Aggarwal et al. 2004) such as domain-specific constraints and user preferences.

Current approaches match the static behaviour of Web services in terms of whether the service has similar description for functional and non-functional parameters. While for the candidate Web services it is highly likely that these parameters are semantically similar, it is the execution values for such functional and non-functional parameters that provide valuable guidance for decision-making process regarding service adequacy for the task. This is because service behaviour is difficult to presume prior to service execution and can only be formed based on the experience with the service execution. The accuracy of automatic matchmaking of Web services can be further improved by taking into account the adequacy of past matchmaking experiences for the requested task.

Hence, there is a need for a methodology that uses domain-specific knowledge representation system for capturing the Web services execution experiences and reason based on those experiences. Case Based Reasoning (CBR) provides such methodology as its fundamental principle is that experience formed in solving a problem situation can be applied for other similar problem situation. In this paper, we present such a framework based on Semantic Case Based Reasoning (S- CBR), in which reasoning for service discovery and matchmaking is done based on set of previous experiences or cases.

The paper begins with an introduction to Case Based Reasoning. In section 3, we introduce the Semantic CBR modelling applied to Web services. In section 4 we discuss the design of our matchmaking algorithm. Section 5 presents our implementation of such Semantic CBR system for Web services. In section 6 we discuss differences and similarities with other related work. Section 7 presents conclusions and a note

about how we plan to extend this framework in the future.

2. OVERVIEW OF CASE BASED REASONING

We adopted CBR as the engine for our Web services discovery mechanism because CBR's fundamental premise that situations recur with regularity, (Aamodt and Plaza. 1994) i.e. experience involved in solving a problem situation can be applied or can be used as guide to solve other contextually similar problem situation. Reasoner based on CBR hence matches the previous experiences to inspire a solution for new problems. There are three main stages in CBR reasoning:

Case Representation

Case- the core component of CBR system can be defined as a contextualized piece of knowledge representing an experience (Aamodt and Plaza. 1994). Case records knowledge at an operational level by capturing experiences in terms of problem situation and the solution involved. Case vocabularies are the labels or the representation schemas defining what knowledge needs to be captured. These vocabularies need to be organized in modular or structured fashion to make them recognizable by the CBR reasoner; hence various representation styles for case representation exist.

Case Storage and Indexing

Case worthy of storage contributes to the reasoning of CBR reasoner in solving new problem situations. Such Cases need to be indexed and stored in the case library or case base, so that reasoner can retrieve them for reasoning. Apart from efficiency, the purpose of indexing cases is relevance, i.e. to retrieve contextually relevant cases to the new problem.

Case Search and Evaluation

Whenever a new problem needs to be solved, case library is searched for the cases that can provide potential solution. The first phase of the search is *case retrieval*, and uses indexing to retrieve cases that are contextually similar to the new problem. In the next phase *matchmaking,* the retrieved contextually similar cases are further matched or investigated to verify if the possible solution is the one applied to the prior problem situations. If the system does not find an adequate match, then the combined contextual knowledge of relevant cases is applied to solve the problem, this phase is called *adaptation*. On success, adopted cases are entered in the case library. On failure, the situation leading to failure is entered in the case library, which serves as a case and guides the CBR reasoner to avoid future failures in similar situation. Inconsistencies encountered during the evaluation are recorded as cases and are termed *case revision*.

In our approach, Web services execution experiences are modelled as cases. The cases are the functional and non-functional domain specific Web services properties described using semantics. In this modelling, the case library will be the storage place for such execution experiences and is identical to Web service registry in that it stores Web services references, but unlike registries case libraries also describe execution behaviour.

Case retrieval is similar to Web services discovery problem in that both mechanisms seek to find potential Web services for the current problem. *Case matchmaking* is similar to Web services matchmaking as both attempts to select acceptable Web services, from the retrieved Web services during the case retrieval or Web service discovery phase respectively. Case adaptation, applicable when the available cases cannot fulfil the problem requirements by and carried out by adapting available cases, is similar to Web service composition, as the composition is applied when available services are not sufficient in meeting the requirement for the problem. .

3. USE OF CASE BASED REASONING FOR WEB SERVICES MATCHMAKING

The Framework Architecture

In our S-CBR framework, there are two main roles: *case administrator* who is responsible for case library maintenance by entering or deleting cases from the library and c*ase requestor* who searches the case library to find solution for the problem and is similar in role with Web service requestor. Figure 1 is the schematic diagram for our framework.

S-CBR allows the Web service requestor to provide problem description and search for Web service that meets the requirements. The dynamics of the framework operation is as follows:

1) Initially, the administrator populates the repository with semantic case representation formats for specific application domain. This representation is used to semantically annotate both the user requests for suitable services and the execution experiences of Web services for the specific domain.

2) The S-CBR Engine is the first contact point for the web service requestor, who can use the user interface to input the problem requirements and as a final result receives Web service references with other details. After receiving the problem description, S-CBR starts search for finding suitable services that matches the request.

3) At this stage, the CBR Engine passes new problem description and the custom semantic case representation format to the Semantic Description generator module, which annotates the new problem according to the representation format

4) The annotated problem is then passed to the *indexing* module, which computes the suitable index for the new problem and passes the index to the *Case retrieval* module. .

5) The c*ase retrieval* module queries the case library for cases with the similar indexes. Output at this stage will be the cases, which have similar index to the current problem and these retrieved cases are passed to the next stage.

6) The *case matchmaking module* takes retrieved cases and the annotation of problem description from the *semantic description generator* module, and outputs matched cases.

7) The CBR engine receives these matched cases and extracts the Web services details from the solution part of the case.

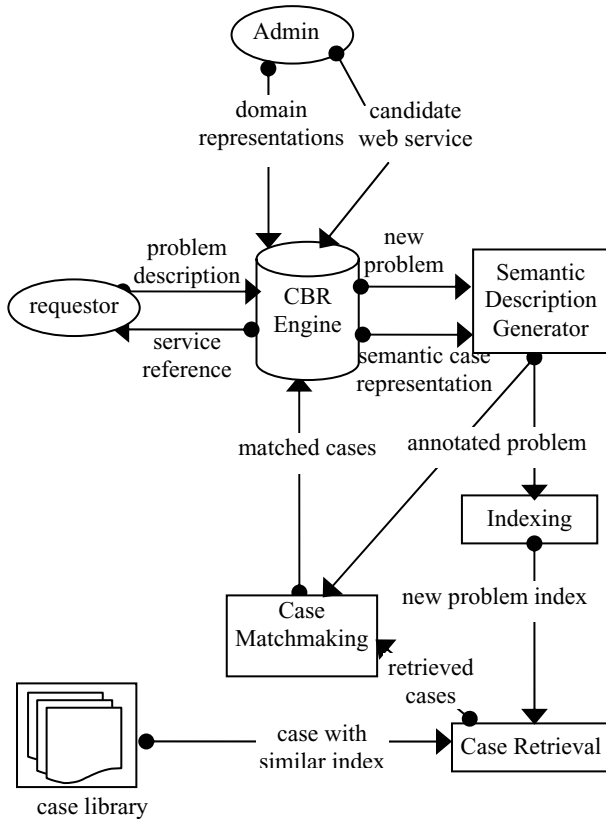8) The CBR engine returns Web services details to the service requestor.

Figure 1: Architecture of the S-CBR framework

Although the chosen case study for this work is from the travel domain, the modular, ontology-driven design of framework makes it application-independent and allows its seamless reuse for other application domain.

Semantics for Case Representation and Storage

This section provides details on the CBR modelling to address the Web services discovery and matchmaking for specific application domains, exemplified by the classic Travel Domain problem (Thakker et al. 2005). In this problem domain the user (Web service requestor) wants to find suitable Web service for a planned travel trip.

Case Vocabularies

In our approach, we store Web services execution experiences as cases. The vocabulary for such cases includes functional and non-functional parameters, which characterize the Web services execution behaviour. The choice of these parameters should be

domain-specific and should represent the knowledge associated with the context of that particular domain.

In our travel domain based framework the vocabulary includes functional and non-functional parameters:

1) The functional parameters are inputs (i.e. travel details) to the travel domain Web services and outputs (i.e. travel itinerary) from such Web services.

2) Non-functional parameters are constraints on various values (i.e. exclusion of particular travel medium or particular travel medium instance) or preference for certain parameters (i.e. price range, currency, Quality of Service expected). In addition, execution experiences stored in the case library should also include the solution and a notion to specify if the solution was acceptable for the end-user. Features that characterise the domain are extremely useful for top-level indexing and can also be included as non-functional parameters.

Case Representation using Frame structures.

After deciding on the knowledge and corresponding vocabulary to be represented as a case, we need to decide how this knowledge can be represented.

In our approach, we adopt frame structures for the case representation. In frame structures, *frame* is the highest representation element consisting of *slots* and *fillers*. *Slots* have *dimensions* that represent lower level elements of the *frame*, while *fillers* are the value range the *slot dimensions* can draw from. In our implementation, *slot dimensions* represent case vocabulary in modular fashion while *fillers* describe the possible value ranges for the *slot dimensions*.

Frame representations are highly structured and modular which allows handling complexity involved in representation. Moreover, frame structure has relationship with semantic net, which makes natural transition to the Semantic Web descriptions possible. Table 1 shows such a frame structure for our travel domain case vocabulary.

Table 1: Travel Domain Frame Structure

Slot	Dimension	Filler
Travel Request	City Departure	valid city
	City Arrival	valid city
Constraints on Goal	On Instance	Valid Travel Domain Instance
	On Domain	Valid Travel Domain
Preferences	On Price range	positive Double
	On Currency	valid currency
	On QoS parameter	possible QoS parameter(s)
Features	Travel Regions	Domestic/International

Mapping Frame structure to Ontologies.

In the developed framework, we map the frame structures to ontologies. We derive rules for such mapping as described in Figure 2. According to this mapping, *frame* and *slot* are represented as classes. The

relationship between *frame* and *slot* is expressed in terms of properties of *frame*, as the range for these properties are the *slot* classes. *Dimensions* are the properties of the *slots*. Possible range for these properties is the values the respective *filler* can derive from.

We use Web Ontology Language (OWL), a Semantic Web standard for constructing these ontologies. OWL is the most expressive Semantic Web knowledge representation so far. The layered approach adopted by semantic web, allows reasoning and inference based on ontologies, which is the most powerful and ubiquitous feature of Semantic Web. After applying the mapping, the ontology for the travel domain case representation is created (Figure 2), where CaseRepresentation class has: *hasTravelRequest*, *hasTravelResponse*, *hasConstraintsOnGoal*, *hasPreferences*, *hasFeatures*, *hasSolution* and *hasFeedback* object properties. Range for these properties are TravelRequest, TravelResponse, Constraints, Preferences, Features, Solution, and Experience classes respectively.

(S-CBR. 2006) contains the main ontology classes that CaseRepresentation ontology refers to in our framework, namely TravelRequest, Constraints, Features, Solution, Feedback, QoS and City. In order to exercise the noble objective of globalization of semantic descriptions, we used external ontologies where appropriate (Currency 2005, Portal 2005). For instance, the cityOfArrival is an object property referring to the publicly available ontology (Portal 2005) where other useful information about the specific city can be found such as country, the number of inhabitants, etc.
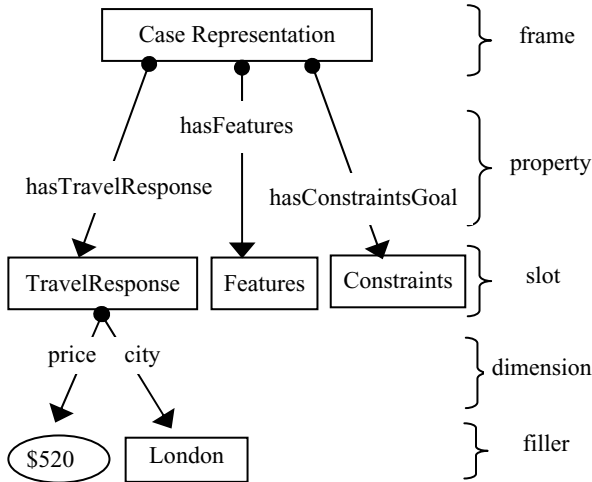

Figure 2: Mapping between frame structure and semantic case representation for Travel Domain

After modelling cases in OWL based semantic descriptions, it is possible to reason using OWL reasoner (Parsia and Sirin 2004). Each new case stored in the case library, will be an instance of the ontology class CaseRepresentation. This makes it possible to derive inference for the purpose of decision-making, which involves further phases of CBR system.

Frame structure based case representation for Web services execution experiences, which has a notion for describing functional and non-functional parameters, provides a mechanism to represent higher structured real-life problems. For instance, a real world web services execution problem described in plain English representation: "Find a Trip for single person, Mr Lee; Mr Lee wants to travel from Boston to New York, with price range in total $220, He does not want to travel by road. The dates of Travel will be 27-02-2005 for departure and 01-03-2005 as return date. He prefers to pay in USD. He needs quick results (approximately in 1.5 seconds)." will be transformed as frame as shown in Table 2.

Table 2: Example of a Case

Travel Request	
City Departure	New York ([City (USA [Country])]) is-city-of
City Arrival	Boston ([City (USA [Country])]) is-city-of
Preferences	
On Price range	220
On Currency	USD ([Currency]) code
Features	
Travel Regions	Domestic ([Travel Regions])
Class = [class], Instance = instance ([class]), Property = properties	

4. S-CBR FRAMEWORK DEVELOPMENT

Case Indexing and Storage.

Cases can be indexed based on vocabularies, which should allow retrieval of appropriate cases during the search procedure. For indexing cases in our framework, we use "partitioning the case library" method, which is a variation of "flat memory indexing" technique (Kolodner 1993). In this indexing method, case library is partitioned based on certain vocabularies and the new problem is recognized based on the identical vocabularies to decide which partition the problem falls into. In our architecture, cases are stored based on vocabulary element *Features* as presented in Table 1, which corresponds to *hasFeatures* property (see Figure 2) from the CaseRepresentation ontology class. For our travel agent case study, the possible values for this vocabulary (*hasFeatures* property) are either *Domestic* or *International* (pre-defined instances from the TravelRegion class), hence indexing will partition case library into two parts. This simple procedure for indexing works for our proof-of-concept, however the real-world Case Based Reasoning system can use more than one vocabulary term or combinations of vocabulary terms for indexing.

Case Retrieval

Whenever a new Web service needs to be searched, the problem description involving the functional parameters and non-functional parameters are encoded

using the case representation frame structure. In our framework, the new problem description is an instance of *CaseRepresentation* ontology, which involves possible values for properties: *hasTravelRequest*, *hasConstraintsOnGoal*, *hasPreferences* and *hasFeatures.*

Our CBR architecture identifies the new problem based on the partition it falls into, and then the rest of the matching is applied to cases from that partition only. In Framework, this corresponds to using *hasFeatures* property value to reason whether the new problem falls under *Domestic* or *International* Travel Region. Based on the outcome of reasoning, the cases associated with particular partition are further investigated.

Case Matchmaking and Ranking

The case retrieval procedure suggests cases that are a potential solution to the problem. The matching procedure that narrows down the retrieved cases to present acceptable solution(s). From the available methods for matchmaking in CBR literature, we choose Nearest-Neighbour Matching and Ranking using numeric evaluation function (ReMind 1992) method for our framework. This method assigns importance ranking to properties of case and then computes the degree-of-match by comparing the cases based on values for these properties. Applying this procedure to all the retrieved cases gives the best matching cases. We implement the Nearest-Neighbour Matching and Ranking method in the following manner:

We rank case dimensions that contribute to the decision-making procedure for finding a successful solution. Ranking is applied to the *slot dimensions* with the importance indicator in the range of 0-1, where 1 indicates the highest importance. For example, we assign *City Departure* and *City Arrival* importance of 1.0.

Next, for the new problem and each retrieved cases we compare the *filler* values for ranked *dimensions* to derive the degree of match. As the implementation of case is in terms of semantic descriptions, the degree of match is semantic based. To find degree of match, we use two different approaches for *dimensions* implemented as object properties and those implemented as data type property.

Object Property Comparisons

For the *dimensions* annotated as object properties, the possible *filler* values will be an instance of *slot* class. Hence, for semantically matching object property value for the new problem and the retrieved cases, the algorithm compares the instances. If the instances matches then the degree of match = 1, otherwise the algorithm traverses back to the super class of the class, the instance is derived from and comparison is performed at that level.

The comparison is similar to traversing tree structure, where the tree representing the class hierarchy for the ontology element. This procedure of traversing back to the super class and matching instances is

repeated until there are no super classes in the class hierarchy; hence leaf node for the tree is reached, giving degree of match=0. The value assigned for the partial match will be,

$$DoM = \frac{MN}{GN}$$

Where, MN is the Total number of matching nodes in the selected traversal path, and GN Total number of nodes in the selected traversal path.

For example, comparing the departureCity object property values for the new problem request and the retrieved cases (#1 and #2) will give degree of match for Case#1 =(0/3)=0 because no matches are found while traversing the ontology tree until the leaf node is reached. However for Case#2, the degree of match will be 2/3=0.67 as the instances (New Jersey, New York) does not match but the instance of the Country super-class match. Refer Figure 3
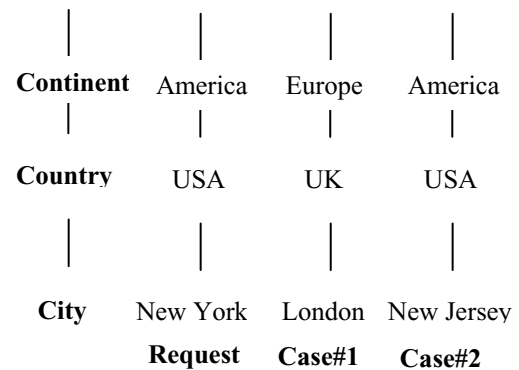
Continent	America	Europe	America
Country	USA	UK	USA
City	New York	London	New Jersey
	Request	**Case#1**	**Case#2**

Figure 3: Semantically matching object properties

DataType Property Comparisons

For semantically comparing the dimensions that are implemented as data type properties, we use qualitative regions based measurement (Kolodner 1993) method. In this method, the qualitative range classifies possible values into number of qualitative regions. The respective values in the retrieved case and new problem are evaluated based on which region they fall into. Degree of match =1 if they fall under the same qualitative region. Otherwise, degree of match is assigned based on the relative distance from the new problem's qualitative region. Higher the distance, lower the value. Hence,

$$DoM = 1 - \left\{ \frac{DQ}{TQ} \right\}$$

Where, DQ is the Distance from the desired qualitative region and TQ is the Total number of the qualitative regions. For example, if the hasPreferenceOnpriceRange data type property have qualitative regions (0-299), (300-599) and over 600. Comparing price range for the new problem request (350) with the respective retrieved case (220): value for the retrieved case falls under range (0-299), while for

the new problem it is (300-599), hence the degree of match is (1- 1/3)=0.66.

Computing the overall similarity value

Apply following evaluation function (ReMind 1992) to compute the aggregate degree of match for each retrieved case.

$$ADoM = \frac{\sum_{i=1}^{n} W_i * sim\ (f_i^N, f_i^R)}{\sum_{i=1}^{n} W_i}$$

Where, n is the number of ranked dimensions, W_i is the importance of dimension i, *sim* is the similarity function for primitives, and f_i^N and f_i^R are the values for feature f_i in the new problem and the retrieved case respectively. The evaluation function sums the degree of match for all the dimensions as computed in previous step, and takes aggregate of this sum by considering the importance of dimensions.

The accuracy of Web services discovery and matchmaking is dependent on the right combination of indexing, ranking and the existence of adequate cases in the case library.

5. IMPLEMENTATION HIGLIGHTS

The implementation of our framework uses semantics extensively to implement the component of the CBR system as case representation is in terms of ontologies, the individual cases are implemented as ontology instances and the CBR system logic for indexing, matching, retrieval and ranking of cases is developed using Java based ontology reasoner. OWL DL was our ontology language of choice. We used Pellet - a Java based OWL reasoner.
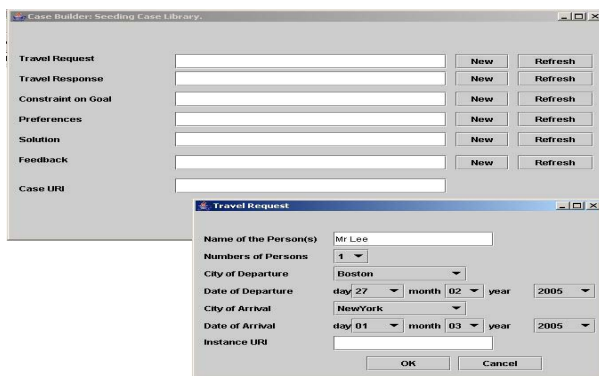

Figure 4: User Interface

Figure 4 illustrates a snapshot of the GUI developed for the matchmaking framework. The interface allows different options to two kinds of users: The case administrator, who is responsible for case library maintenance and a case requestor who wants to retrieve Web service for a trip. The implementation issues case administrator privileges for administrator in order to perform case maintenance activities: case seeding, rankings, setting up the threshold value (the acceptable

value for matching coefficient) and deletion of old cases. Case requestor can also setup rankings, which will be applicable for a particular session.

While seeding the case library with a new case, the interface assists the case administrator in creating the ontology instances. The main feature of the framework is that the program creating the user Interface uses *CaseRepresentation* class from the CaseRepresentation (Figure 2) ontology to form the GUI elements. Subsequent properties from the CaseRepresentation class and the range for those properties constitute the rest of the user interface. For example, one GUI component in figure 3 shows the mode in which case administrator is assisted for creating the instance of Travel Request class while entering *hasTravelRequest* property of CaseRepresentation class. The value entered for particular property is validated against the range and cardinality from the ontologies. Framework also makes the possible instances available once they are created. For example, in Figure 4 while entering values for the Travel Request, city instances *Boston* and *New York* are available for re-use. As a result of seeding a new case, framework creates an ontology instance of *CaseRepresentation* class and stores into case library.

For case searching, the framework assists the *case requestor* with the user interface similar to that available for case administrator, and creates semantic description for the new problem parameters. Generated index for such semantically described problem governs the decision regarding which partition the problem falls into and the cases from that partition are retrieved for further matching. This matching procedure is implementation of the algorithm described in previous section. Result of the matching procedure displays the case instances, which have similar problem situation to the new problem. The framework also displays the aggregate matching coefficient associated with such suggested case instances for the *case requestor* to view and make appropriate selection.

6. RELATED WORK

Semantic descriptions are increasingly being used for exploring the automation features related to Web services discovery, matchmaking and composition. The semantic approach to composition aims to achieve a more dynamic composition by describing the Web services semantically, thus allowing software agents to reason about the service capability, and make all the decisions related to the composition on behalf of the user or developer. OWL-S (Ontology Web Language for Web services) is the leading semantic composition research effort. OWL-S (Dean et al. 2005) ontologies provide a mechanism to describe the Web services functionality in machine-understandable form, making it possible to discover, and integrate Web services automatically. An OWL-based dynamic composition approach is described in (Sirin et al 2003), where semantic description of the services are used to find matching services to the user requirements at each step of composition, and the generated composition is then

directly executable through the grounding of the services. Other Approaches use Artificial Intelligence planning technique to build a task list to achieve composition objectives: selection of services and flow management for performing composition of services to match user preferences. A constraint driven composition framework in (Aggarwal et al 2004), uses functional and data semantics with QoS specifications for selecting Web services. These approaches use semantics for automatic Web services discovery; however overlook the Web service execution behaviour in decision-making.

Experience based learning using CBR is a relatively old branch of Artificial Intelligence and Cognitive Science and is being used (Hammond 1986) as an alternative to rule-based expert system for the problem domains, which have knowledge captured in terms of experiences rather than rules. However, Case based reasoning for Web services were initially documented in (Limthanmaphon and Zhang 2003), where the developed framework uses CBR for Web services composition. In their approach, the algorithm for Web services discovery and matchmaking is keyword based and has no notion for semantics. This affects the automation aspects for Web services search and later for composition. Our framework consumes semantics extensively and achieves the automation required for web service discovery and matchmaking. Use of ontologies also makes our framework extensible and reusable.

7. CONCLUSIONS AND FUTURE WORK

Semantic description of Web service profile paves the way for automating the discovery and matchmaking of services since it allows intelligent agents to reason about the service parameters and capabilities. However, the accuracy of such automatic search mechanism largely relies on how soundly formal methods working on such semantic descriptions consume them.

In this paper, we argued for the importance of considering the execution values for semantically described functional and non-functional Web services parameters in decision making regarding Web service adequacy for the task. This is because the service behaviour is impossible to presume prior execution and can be only generalized if such execution values are stored and reasoned for deciding service capability. AI planning and Intelligent Agent based reasoning methods provide rule-based reasoning methodology rather than experience-based. Hence, we used CBR method that allows capturing experiences and reasoning based on them. We implemented Semantic Case based Reasoner, which captures Web service execution experiences as cases and uses these cases for finding a solution for new problems. The implemented framework extensively uses ontologies, as semantics are used for describing the problem parameters and for implementing components of CBR system: representation, indexing, storage, matching and retrieval. These components are modelled based on ontologies, making the application logic

captured within semantic descriptions. Semantic approach for modelling CBR reasoner achieves required automation and makes CBR reasoner extensible and reusable.

A problem that research in semantic-based matchmaking and composition has not addressed sufficiently is the interoperation between independently developed reasoning engines. Without this interoperation, the reasoning engines remain imprisoned within their own framework, which is a drawback, especially that most engines usually specialize in serving a particular domain, hence interoperation can facilitate inter-domain orchestration. We believe that in this work we took a small step forwards standardization at the reasoner level by describing the CBR reasoning model semantically.

Future work will involve exploring case adaptation, which is applicable when the available cases cannot fulfil the problem requirements, so matchmaking is attempted by adapting available cases.

REFERENCES

Aamodt , A., and Plaza, E. 1994. Case-based reasoning: foundational issues, methodological variations, and system approaches. *AI Communications* 7, No.1, 39-59.

Aggarwal, R., Verma, K., Miller, J.A. and Milnor, W. 2004. Constraint Driven Web Service Composition in METEOR-S. *Proceedings of the 2004 IEEE International Conference on Services Computing (SCC 2004)*, Shanghai, China, 23-30.

Currency.2005. www.daml.ecs.soton.ac.uk/ont/currency.daml

Dean, M., Hendler, J., Horrocks, I., McGuinness, D., Patel-Schneider, P. F., and Stein, L. A.. 2005. Semantic Markup for Web Services, OWL-S version 1.1. Retrieved April 10, 2006 from http://www.daml.org/services/owl-s/1.1/

Hammond, K. 1986. Learning to anticipate and avoid planning problems through the explanation of failures. *In proceedings of AAAI-86*. Cambridge, MA: AAAI press/MIT press.

Kolodner, J., 1993. Case-Based Reasoning. Morgan Kaufmann Publishers Inc. ISBN:1-55860-237-2.

Limthanmaphon, B., and Zhang, Y. 2003. Web service composition with case-based reasoning, *Proceedings of the Fourteenth Australasian database conference on Database technologies*, Adelaide, Australia, 201 – 208.

Parsia, B., and Sirin, E. 2004. Pellet: An OWL DL Reasoner. *In Third International Semantic Web Conference (ISWC2004)*, Hiroshima, Japan.

Portal. 2005. http://www.aktors.org/ontology/portal

ReMind 1992. Developer's Reference Manual, Cognitive Systems, Boston.

S-CBR. 2006. project Web site: http://semweb.i8.com/SCBR.htm

Sirin, E., Hendler, J., and Parsia, B. 2003. Semi-automatic composition of web services using semantic descriptions. *In Web Services: Modelling, Architecture and Infrastructure workshop in ICEIS 2003*, Angers, France.

Thakker, D., Osman, T., and Al-Dabass, D. 2005. Bridging the Gap between Workflow and Semantic-based Web services Composition, In the *Proceedings of the Workshop on WWW Service Composition with Semantic Web Services 2005*, Compiègne, France, 13-23.