Utilisation of Case-Based Reasoning for Semantic Web Services Composition

Taha Osman, Nottingham Trent University, UK

Dhavalkumar Thakker, Nottingham Trent University, UK

David Al-Dabass, Nottingham Trent University, UK

ABSTRACT

With the rapid proliferation of Web services as the medium of choice to securely publish application services beyond the firewall, the importance of accurate, yet flexible matchmaking of similar services gains importance both for the human user and for dynamic composition engines. In this article, we present a novel approach that utilizes the case based reasoning methodology for modelling dynamic Web service discovery and matchmaking, and investigate the use of case adaptation for service composition. Our framework considers Web services execution experiences in the decision making process and is highly adaptable to the service requester constraints. The framework also utilizes OWL semantic descriptions extensively for implementing both the components of the CBR engine and the matchmaking profile of the Web services.

Keywords: *case based reasoning (CBR); matchmaking and composition; Semantic Web; Web services*

INTRODUCTION

The Internet has become the market-place for a colossal variety of information, recreational and business services. Web services are increasingly becoming the implementation platform of choice to securely expose services beyond the firewall. Moreover, multiple Web services can be integrated either to provide a new, value-added service to the end-user or to facilitate co-operation between various business partners. This integration of Web services is called "Web services composition" and is feasible to achieve because of the Web services advantages of being platform, language neutral and loosely coupled.

Automatic Web service discovery and matchmaking is the principal aspect for dynamic services composition. The accuracy of the matchmaking (selection) process enhances the possibility of successful composition, eventually satisfying the user and application requirements. The current standard for Web

service discovery, the Universal Description, Discovery and Integration (UDDI) registry is syntactical and has no scope for automatic discovery of Web services. Hence, current approaches attempting to automate the discovery and matchmaking process apply semantics to the service descriptions. These semantics are interpretable by the service (software) agents and should include WSDL-based functional parameters such as the Web services input-outputs (Martin et al., 2004a)(Akkiraju et al., 2005), and non-functional parameters such as domain-specific constraints and user preferences (Aggarwal, Verma, Miller, & Milnor, 2004).

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, which gives us valuable information about the services behaviour that is difficult to presume prior to service execution. Hence, there is a need for a methodology that uses domain-specific knowledge representation of the required task to capture the Web services execution experiences and utilise them in the matchmaking process. Case Based Reasoning (CBR) provides such methodology as its fundamental premise is that experience formed in solving a problem situation can be applied for other similar problem situation.

The article begins with describing the motivation behind the work. In the following section we review theory of Case Based Reasoning and describe how it can be utilised for modelling Web services matchmaking. Next we discuss the design of our matchmaking algorithm, its implementation highlights, and analyze preliminary results. Finally we investigate how case adaptation can further extend our matchmaking algorithm to cater for service composition and review related work.

MOTIVATION

The most practically deployed Web services composition techniques use the theory of business workflow-management as composition process model to achieve formalization for control and data flow. Mainly based on the Business Process Execution Language (BPEL) standard (Andrews et al., 2003), these techniques also have practical capabilities that fulfil the needs of the business environment, such as fault handling and state management. However, the main shortcoming of these techniques is the static selection and composition approach, where the service selection and flow management are done a priori and manually.

A popular research direction attempts to improve BPEL composition by introducing semantics to workflow-based composition (Osman, Thakker, & Al-Dabass, 2005). However, these approaches also 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 the decision-making process regarding the 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.

Hence, the problem requires a methodology, which has the domain-specific knowledge representation system for capturing the Web services execution experiences and reason based on those experiences. We adopted CBR (Case Based Reasoning) as the engine for our Web services discovery mechanism because CBR's fundamental premise that situations recur with regularity (Aamodt & 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 situations. Reasoner based on CBR hence matches the previous experiences to inspire a solution for new problems.

OVERVIEW OF CASE BASED REASONING

The Case-Based Reasoning technology was developed in 1977 based on the research effort of Schank and Abelson. They proposed that our general knowledge about situations is recorded in the brain as scripts that allow us to set up expectations and perform inferences (Watson, 1997) The processes involved in CBR can be represented by a schematic cycle comprising four phases (Aamodt et al., 1994):

- RETRIEVE the most similar case(s);
- REUSE the case(s) to attempt to solve the problem;
- REVISE the proposed solution if necessary, and
- RETAIN the new solution as a part of a new case.

There are 4 main stages in CBR reasoning:

Case Representation

A case is a contextualised piece of knowledge representing an experience (Aamodt et al., 1994). It contains the problem, a description of the state of the world when the case occurred, and the solution to this problem. The solution contains elements that address the problem and inform about the relevance of the solution. When a reasoner is created, the elements of the case are defined according to the context. For example, the city of departure or the number of passengers could be some elements to represent a travel experience as a case. Case vocabularies are thus developed for each reasoner, to define what knowledge needs to be captured.

Case Storage and Indexing

Cases are then stored in a case library or case base. It is an important aspect for the designing of CBR systems because it reflects the conceptual view of what is represented in the case. The structure of the library should permit efficient search by the reasoner. This search can be facilitated by the use of indexing. Indices are therefore assigned to cases in order to express information about the case content.

Case Retrieval

Whenever a new problem needs to be solved, the case library is searched for the cases which can be a potential solution. The first phase of this search is case retrieval, which aims to find cases that are contextually similar to the new problem. The retrieval is done according to the index of the cases.

Matchmaking

Matchmaking performs the comparison between retrieved cases and the request to verify if a past solution can be reapplied. There are several available methods for matchmaking in CBR literature. The Nearest-Neighbour Matching and Ranking is an interesting one because it involves the assessment of similarity between stored cases and the input (request) case. It assigns importance ranking to properties of cases and then computes the degree of matching by comparing the cases for these properties (Kolodner & Simpson, 1989). The matchmaking process is thus performed on each retrieved case, and the most similar case to the input case is assigned the highest ranking. If the system finds a matching case, it is possible to reuse the solution suggested by the retrieved case for the new problem.

In our CBR matchmaking 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.

The apparent compatibility confirms our thesis that the CBR methodology is well suited to build automatic Web service composition frameworks

MATCHMAKING WEB SERVICES USING CASE BASED REASONING

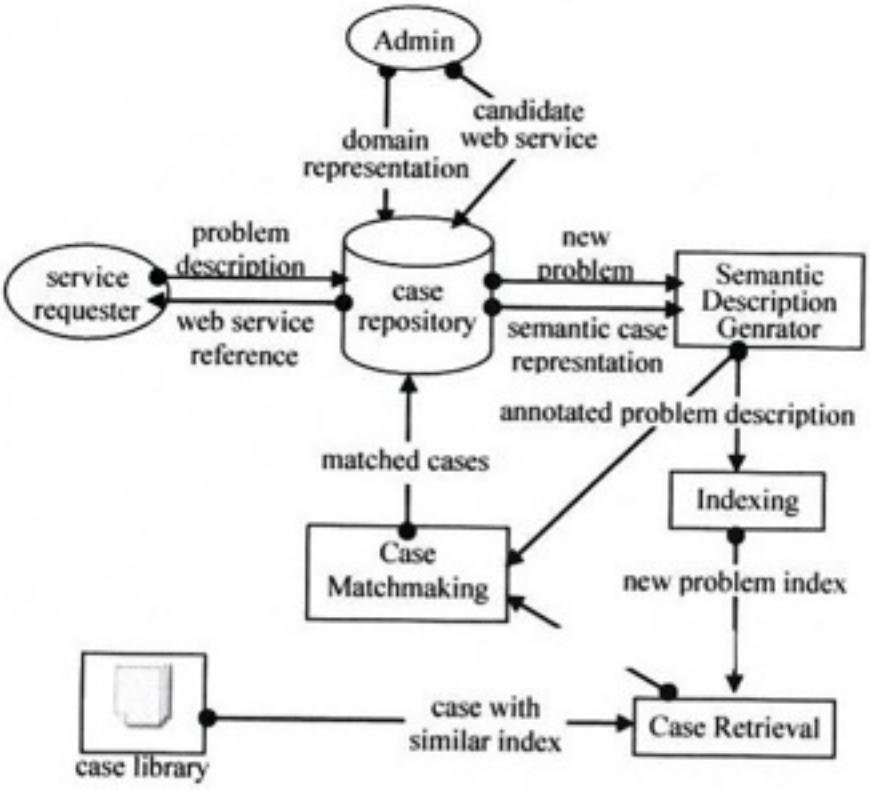
The Framework Architecture

In our Semantic CBR matchmaking, there are two main roles: case administrator who is responsible for case library maintenance by entering or deleting cases from the library and case requestor who searches the case library to find solution for the problem. Figure 1 illustrates a schematic diagram for our framework.

The dynamics of the framework operation is as follows:

- 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.
- The user inputs the service requirements and as a result receives Web service references via the framework interface. The same interface is used by service provides or the system administrator to subscribe a Web service as a candidate for available services for the specific domain.
- The case representation repository retrieves the appropriate semantic case representation format for the requested service and

Figure 1. Architecture of the CBR matchmaking framework



forwards it together with the problem description to the Semantic Description generator module, which semantically annotates the new problem according to the representation format.

- The annotated problem is then passed to the indexing module, which computes a suitable index for the new problem based on the domain feature and/or the functional parameters of the requested service. The index is passed for case retrieval.
- The case retrieval module queries the case library for cases with similar indexes. Output at this stage will be the cases that have similar index to the current problem, which will be candidates for matchmaking.
- The case matchmaking module takes the retrieved cases and the annotation of problem description from the semantic description generator module, runs them through a matchmaking algorithm and forwards the closest match Web service to the requester.

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 for its seamless reuse for other applications domain. In order to enable matchmaking for the financial markets domain for instance, it would suffice to enter a new case representation format into the repository, keeping the rest of the reasoning logic intact.

Ontology Support for Case Representation and Storage

The most common use of ontologies is the reconciliation between syntactically different terms that are semantically equivalent. Applied to CBR case descriptions for Web services, ontologies can be used to provide a generic, reasoner-independent description of their functional and non-functional parameters. Moreover, ontologies can also be used to further index and structure cases with key domain features that increase the efficiency of the matchmaking process. For instance, we can add a feature to the travel domain ontology to indicate whether a trip is domestic or international. Web services QoS parameters are also indexed using ontologies to further improve the accuracy of case matchmaking.

In our framework, ontologies are also used to describe the rules of the CBR reasoning engine, which not only streamlines the intercommunication between the Web service, user request, and the case library, but promotes exploring the collaboration at the reasoning level between different composition frameworks.

Case Vocabulary

In CBR theory, the first step is to define all the elements contained in a case and the associated vocabulary that represents the knowledge associated with the context of a specific domain (our case study is the travel domain).

This vocabulary includes functional and non-functional parameters:

- Functional parameters are the service input (e.g. the travel details) and the service output or results (e.g. the travel itinerary). Input corresponds to the request of the user (e.g. date or city of departure) whereas output corresponds to the response given to the user (e.g. price, flight number).
- Non-functional parameters are constraints imposed by the user (e.g. exclusion of particular travel medium) or preferences over certain specific parameters (e.g. Price range, Quality of Service expected). In addition, execution experiences stored in the case library should also include the solution (i.e. Web services effectively used) and a notion to specify if the solution is 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 (Kolodner et al., 1989) 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.

The frame representations are highly structured and modular which allows handling the complexity involved in representation. Moreover, frame structure has a natural mapping to the semantic OWL description language as the semantic net representations largely borrowed from the frame structures(Elaine & Kevin, 1992), which makes natural transition to the Semantic Web descriptions possible. Table 1 shows such a frame structure for our travel domain case vocabulary.

The slot Travel Request corresponds to the Input, i.e. all the travel details for a travel agent. The Travel Response slot corresponds to the Output, i.e. the answer given to the user at the end of the process. The elements of the answer are the price and the corresponding currency, the access point to the WSDL file of the corresponding Web Services and the Services Used (companies involved in the trip, e.g. an airline and a hotel).

Semantic Encoding of the Frame Structure

In the developed framework, we map the frame structures to ontologies. We derive rules for such mapping as described in Figure 2.

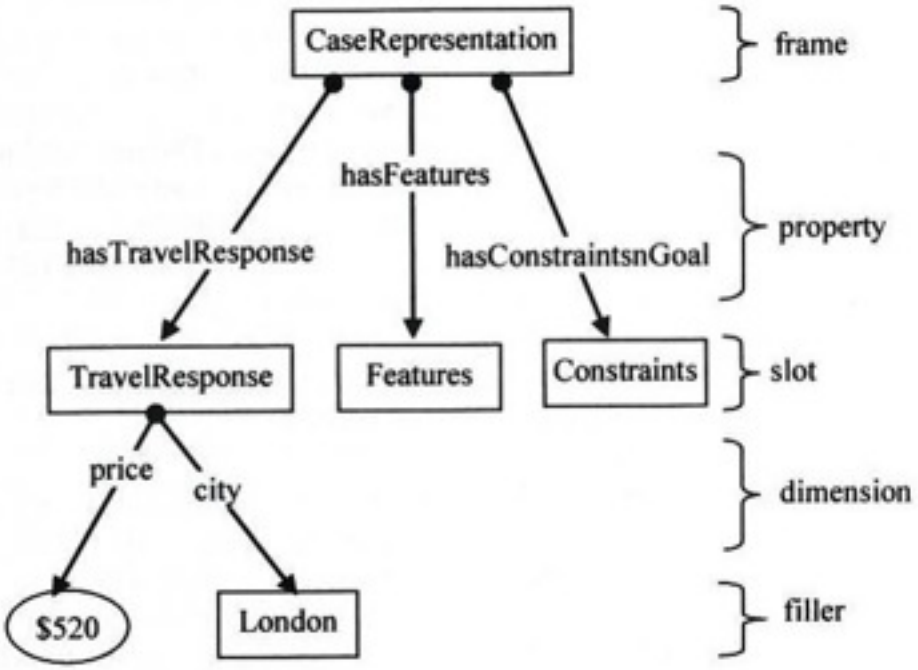
According to this mapping, frame and slot are represented as classes. The relationship between frame and slot is expressed in terms of properties of a frame, i.e. the range for these properties are the slot classes. Dimensions are the properties of the slots. Possible range for these properties is the 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

Table 1. Representation of a case

Slot	Dimension	Filler
Travel Request	Name of Traveler	Any text
	Date of Arrival	Any valid date
	City of Departure	Any valid city
Travel Response	Solution	Service WSDL file
	Price Range	Any positive Double
	Currency	Any valid currency
Constraints	On Domain	Any Valid Travel Domain
	On Price range	Any positive Double
	On QoS parameter	Any possible QoS parameter(s)
Features	Travel Regions	Domestic/International

Figure 2. Mapping between frame structure and semantic case representation



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, where for instance the CaseRepresentation class has: hasTravelResponse, hasConstraintsOnGoal, and hasFeature object properties. Range for these properties are TravelResponse, Constraints, and Feature classes respectively.

In order to exercise the noble objective of globalization of semantic descriptions, we used external ontologies where appropriate. For instance, the property cityOfArrival is an object property referring to a publically available ontology(PORTAL, 2003) , where other useful information about the specific city can be found such as country, the number of inhabitants, etc.

An example of a semantically-encoded travel request is illustrated in Table 2. "Find a Trip for a single person, Mr Lee; Mr Lee wants to travel from Boston to New York, with a maximum price range in total of $220, He does not want to travel by road. The dates of Travel will be 27-02-2005 for departure and 01-03-2005 for return. He prefers to pay in USD and requires a fast response (approximately in 1.5 seconds)".

Case Storage

All the Web service execution experiences, i.e. solutions deemed valid for a particular request, are stored in the Case Library to be reused by the reasoner. The Case Library itself is also an ontology. It contains some instances of the class CaseRepresentation (e.g. a travel experience or a travel case).

DEVELOPMENT OF THE CBR MATCHMAKING FRAMEWORK

Case Indexing and Retrieval

To facilitate the search procedure, cases are indexed based on vocabularies. In our framework, we use "partitioning the case library"

Table 2. Example of a case

Name of passengers	Lee	<TravelRequest:namePassengers>Lee
City of Arrival	Boston	<TravelRequest:cityArrival rdf:resource= "http://localhost/uk/2005/City.owl#Boston"/>
Date of Arrival	1-3-05	<TravelRequest:dateArrival>2005-03-01
Constraint on domain	Road	<Constraints:OnDomain rdf:resource= "http://localhost/ntu/TravelDomain.owl#Airline"/>
Constraint on price	220	<Constraints:OnPrice>220
Constraint on currency	USD	<Constraints:OnCurrency rdf:resource= "http://ecs.soton.ac.uk/currency.daml#USD"/>
Constraint on QoS	1.5 s	<QoS:ExecutionDuration>1.5

method, which is a variation of "flat memory indexing" technique (Kolodner et al., 1989). 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 from the CaseRepresentation ontology class. For our travel agent case study, the possible values for this property are either Domestic or International (predefined instances from the TravelRegion class), hence indexing will partition case library into two parts. In more complex examples more than one vocabulary term or a combination of terms can be used for more sophisticated indexing. As in relational databases selection, the efficiency of the retrieval process largely depends on the precision of the indexing.

Whenever a new Web service needs to be fetched, the problem description involving the functional parameters and non-functional parameters are encoded using the case representation frame structure, i.e. as an instance of CaseRepresentation ontology as illustrated in Table 2.

Matchmaking and Ranking

Case retrieval fetches Web services that are a potential solution to the problem. The matchmaking process 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. The method operates as follows:

- Compare the similarity for each property, between the new problem and the cases retrieved. The method used for comparison depends on the type of the property.
- Quantify the weight of the similarity. A ranking is assigned to each property in accordance with its importance as exemplified in Table 3.

For each case retrieved, the similarity degree is computed and the case with the highest score corresponds to the best-match. Similarity takes values between 0 and 1, which is attributed to each property for each retrieved case. Our similarity comparison method depends on the type of the dimension: data or object.

Table 3. Quantifying the travel domain case dimensions

Slot	Dimension	Importance (0-1)
Travel Request	City Departure	1.0
	City Arrival	1.0
Constraints on Goal	On Instance	0.2
	On Domain	0.8

Data Property Comparison

To compare data type properties, like the price range or the value of QoS (e.g. execution time), we use the qualitative regions based measurement method (Kolodner et al., 1989). The closer the value in a retrieved case is to the value in the request, the higher the similarity coefficient is.

For each data type property, this formula used is: $|Vr - Vc| \leq X.[Vr|$, where V is the value of the property in the request r or in the retrieved case c and X the factor of tolerance. Thus, a factor of tolerance of 0.9 means the value of the retrieved case should be in ±10% region in relation to the value of the request. The optimum tolerance value is determined by the administrator and can be calculated heuristically.

Object Property Comparison

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 of the new problem and the retrieved cases, the algorithm compares the instances. If the instances match, then the degree of match is 1. Otherwise, the algorithm traverses back to the super (upper) class that the instance is derived from and the comparison is performed at that level.

The comparison is similar to traversing a tree structure (Zhang, Arpinar, & Aleman-Meza, 2003), where the tree represents the class hierarchy for the ontology element. The procedure of traversing back to the upper class and matching instances is repeated until there are no super classes in the class hierarchy, i.e. the top node for the tree is reached, giving degree of match equal to 0. The degree of match (DoM) degree is calculated according to the following equation:

$$DoM = \frac{MN}{GN} \quad (1)$$

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

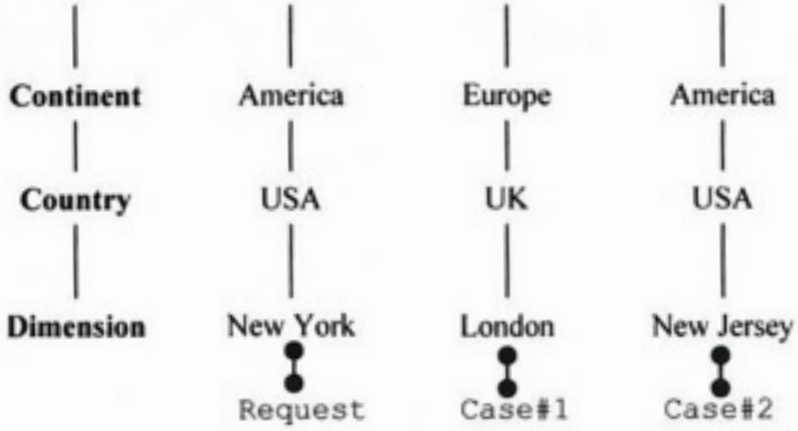
For example, for the request in Figure 3, case#1 will return a degree of match of 0 because no matches are found while traversing the ontology tree until the leaf node is reached. However, for case#2, the degree of match will be 2/3=0.67 as the instances (New Jersey, New York) does not match but the instances of the Country super class match.

It is worth to note that Constraints on object properties are handled by omitting that path in the case ontology tree that renders the constraint invalid. For example, if the passenger is reluctant to travel by air, then the Brit Air, Flight path will not be traversed.

Computing The Overall Similarity Value

Overall similarity is evaluated by computing the aggregate degree of match (ADoM) (Remind, 1992) for each retrieved case according to the following equation:

Figure 3. Semantically matching object properties (dimensions)



$$ADoM = \frac{\sum_{i=1}^{n} W_i \times sim(f_i^N, f_i^R)}{\sum_{i=1}^{n} W_i} \quad (2)$$

Where, n is the number of ranked dimensions, Wi is the importance of dimension i, sim is the similarity function for primitives, and fiN and fiR are the values for feature fi in the new problem and the retrieved case respectively.

The evaluation function sums the degree of match for all the dimensions computed in the previous step, and takes aggregate of this sum by considering the importance of dimensions.

IMPLEMENTATION HIGHLIGHTS

The implementation of our framework uses semantics extensively to implement both the utility ontologies describing the components of the Case-Based Reasoner (Case representation), and the domain ontologies that describe the profile of the Web services in the Case library with a semantic representation (Case Storage).

OWL was our ontology language of choice. We used Pellet (Parsia & Sirin, 2004) - a Java based OWL reasoner, as our ontology engine in favour of the more popular Jena (HP-JENA, 2003), because it supports user-defined simple types. Pellet was used to load and verify (type and cardinality) ontology class instances of user requests and candidate cases.

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 maintaining the case library, and a standard client, who wants to retrieve Web services for a trip. The case administrator has admin privileges to perform case maintenance activities like case seeding, modifying the ranking system or deleting old cases. The client can also setup a ranking system, which will be applicable for a particular session.

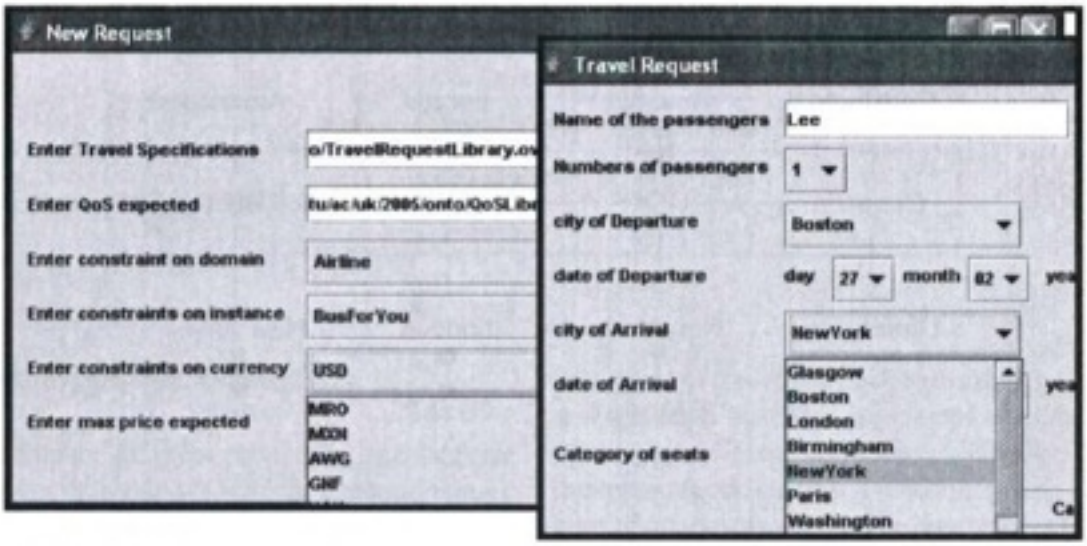
While seeding the case library with a new case or making a new trip request, the interface assists the client in creating the required ontology instances. The value entered for a particular property is validated in relation to the range and cardinality drawn from the ontologies.

The solutions (cases) resulting from the matchmaking process are presented to the client are stored into the case library.

PRELIMINARY RESULTS

At this initial stage of development, the focus of our experiments was to validate the logic of our matchmaking framework, rather than testing a

Figure 4. Admin and user interface



fully working prototype. Hence, we tested our framework with simple in-house developed Web services and compatible wrappers for external publicly available services.

In order to consolidate the test process, we applied different rankings against each test case and associated them with a specific profile. The profile represents a group of users that have similar requirements for the travel request. For instance, the Business profile stands for corporate users, who have to travel frequently; therefore a high standard of comfort is a significant element of choice. These users also need reliability of services. Price is not very important because firms very often have contracts with travel companies. On the other hand, for regular users, represented by the Personal profile, cost is of paramount importance.

The three other types of users are mainly based on specific comparison properties: Economic retrieves cases which price never exceeds a user-defined maximum amount; Travel Medium is specific for constraints on travel domain as well as instances; and Enterprise is useful for companies which are interested in using reliable services. The latter can be important if contracts between the company and different Web services exist so that they can restrict other services.

The rankings are currently administered centrally, but in the future we would like to give the users the opportunity to tweak some of them using a user-friendly interface. Table 4 shows the ranking of our profile system. Example of constraint on Domain is reluctance to travel using a certain transport and constraint on instance the exclusion of certain airline from the search. Quality of service is represented as a single parameter, but in this experiment it is expressed as the availability and response time of the service.

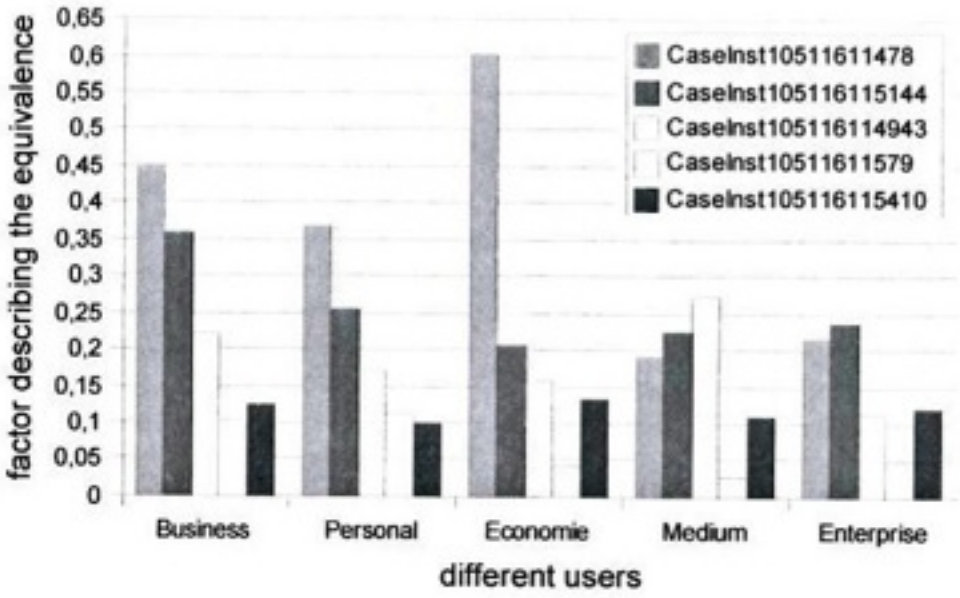
Figure 5 shows the matchmaking degree for different cases using the criteria above. Some cases (Web service execution experiences) present satisfactory results to all users (CaseInst10511611478). Another interesting highlight is that the chosen ranking systems provide different results only if the coefficients are significantly different. This is probably due to the fact that that our case library is not richly populated at the moment.

The average execution time of our matchmaking program at the time of the experiment was approximately 40 seconds, relatively slow considering we only have 30 cases stored in the library. Using semantics has the disadvantage of being more time-consuming than scanning databases. We identified the use of imported

Table 4. User profiles

	Property				
Profile	*Category*	*Constraint on Domain*	*Constraint. on Instance*	*Price*	*Quality of Service*
Business	0.6	0.6	0.4	0.1	0.5
Personal	0.2	0.4	0.7	0.5	0.2
Economic	0.2	0.4	0.2	1	0.1
Travel Medium	0.2	1	0.8	0.3	0.2
Enterprise	0.5	0.3	0.1	0.2	1

Figure 5. User profiles



ontologies as the main performance leak for our program. We plan to develop an off-line caching system to enable us to access the public ontologies locally.

EXTENDING THE MATCHMAKING FRAMEWORK TO WEB SERVICES COMPOSITION

The current framework addresses the problem of automatic Web services discovery and matchmaking by annotating Web services execution experiences and storing them into case base (Osman et al., 2006). The search considers domain-specific criteria and user preferences to find Web services execution experience that solved a similar problem in the past. However the framework assumes that the case library contains suitable cases for every possible problem. This assumption is not always satisfied considering the vast number of problems and problem parameters. Moreover, the framework also needs to deal with situations where the aggregate degree of match (ADoM) is below the *domain-specific* expected degree of match set by the domain administrator or to deal with negative user feedback, where the matched services are not acceptable to the user.

Work under progress involves exploring case adaptation, which is termed as the REVISE phase (Figure 6 - The REVISE phase in CBR) in CBR theory. Adaptation is applicable when the available cases cannot fulfil the problem requirements, so matchmaking is attempted by adapting available cases. Adaptation looks for prominent differences between the retrieved case and the current case and then applies formulae or rules that take those differences into account when suggesting a solution (Watson & Marir, 1994).

Applied to the current framework, when the existing web services experiences in their original form are not sufficient to satisfy current request, the framework should look for relaxing the case restrictions under which a solution is acceptable. If the latter fails, the framework should attempt to merge potential cases to suggest a *composite* solution.

Case adaptation can be defined by the following formula (Maher & Garza, 1997)

$$C' = \alpha(C) \qquad (3)$$

Where, C' = new case, C = old case(s) and α indicates adaptation operator.

The adaptation operator indicates the process of identifying and substituting or transforming an existing solution to fit new situations and is used in *knowledge-based substitution* adaptation.

Knowledge Based Substitutions

In CBR matchmaking process previous cases cannot be always reused without making some changes. Reasoning about these changes requires general and domain specific knowledge to mould case adaptation. For example, if an existing case-solution is applicable for the current travelling problem with the exception of the travel medium – Bus (a road based travel medium which is constrained in the current problem), then the reasoner should use local search on knowledge structure (see the ontology in Figure 7) and conclude the Taxi domain as a possible substitution. Similarly, interpolating parameter of old solution to adapt for the new problem can solve the problem of parameter mismatch.

Another substitution method can use the semantic "sibling" rule for equivalent classes to enables them to replace each other in order to present an appropriate solution.

Under this circumstance, the Equation 3 can be reformulated as:

$$C' = \alpha(C, K) \qquad (4)$$

Where K indicates the influence of general or domain specific knowledge.

The role of knowledge in repairing the existing cases can be described as follows:

- Relaxing the service descriptions (functional parameters) to find a sufficiently

Figure 6. The REVISE phase in CBR

REVISE

IF $ADoM < \{C_{results}, ADoM_{expected}\}$

OR

IF$\{ C_{results}\} \neq U_{expected}$

$U_{expected}$ → User expected results

$C_{results}$ → proposed case solutions

Figure 7. Local search

similar description (applied at the description level).

- Relaxing the execution values of candidate cases (their non-functional parameters) in an attempt to adapt the solution.

The criteria for applying knowledge based substitution are:

- In a situation, where the reasoner cannot find an exact or reasonable match or where exact match is not possible or not desired or not required.
- To make absolutely sure that only possible solution is transformation, which is an expensive operation involving an AI planner, which is a resource expensive exercise.

Hence, if for the current problem C, the available cases in the case library are:

$$C_{library} = \{ C_1 (S_1+S_2, F_1), C_2 (S_2+S_3, F_2), C_3 (S_1,F_3), C_4 (S_2,F_4) , C_5 (S_1+S_2, F_5) \}$$

Where, C (S, F) indicates cases with Web services as solution S applied under circumstances defined by F The circumstances can be characterized by service description, problem description, constraints and preferences applied while solving the problem.

Then the solution for new problem $C, C_6 (S_1 + S_2, F_6)$ must be reached by exploring the matching cases C_1 and C_5 first, before transforming C_3 and C_4 to find a solution from a scratch.

Planning Based Transformations

The planning based transformations can be applicable when the available solutions can not fulfil the problem requirements with normal matchmaking and discovery mechanism or by applying minor modifications using substitution based transformation. Under these circumstances, the equation 3 can be reformulated as:

$$C' = \alpha(C, \rho) \quad (5)$$

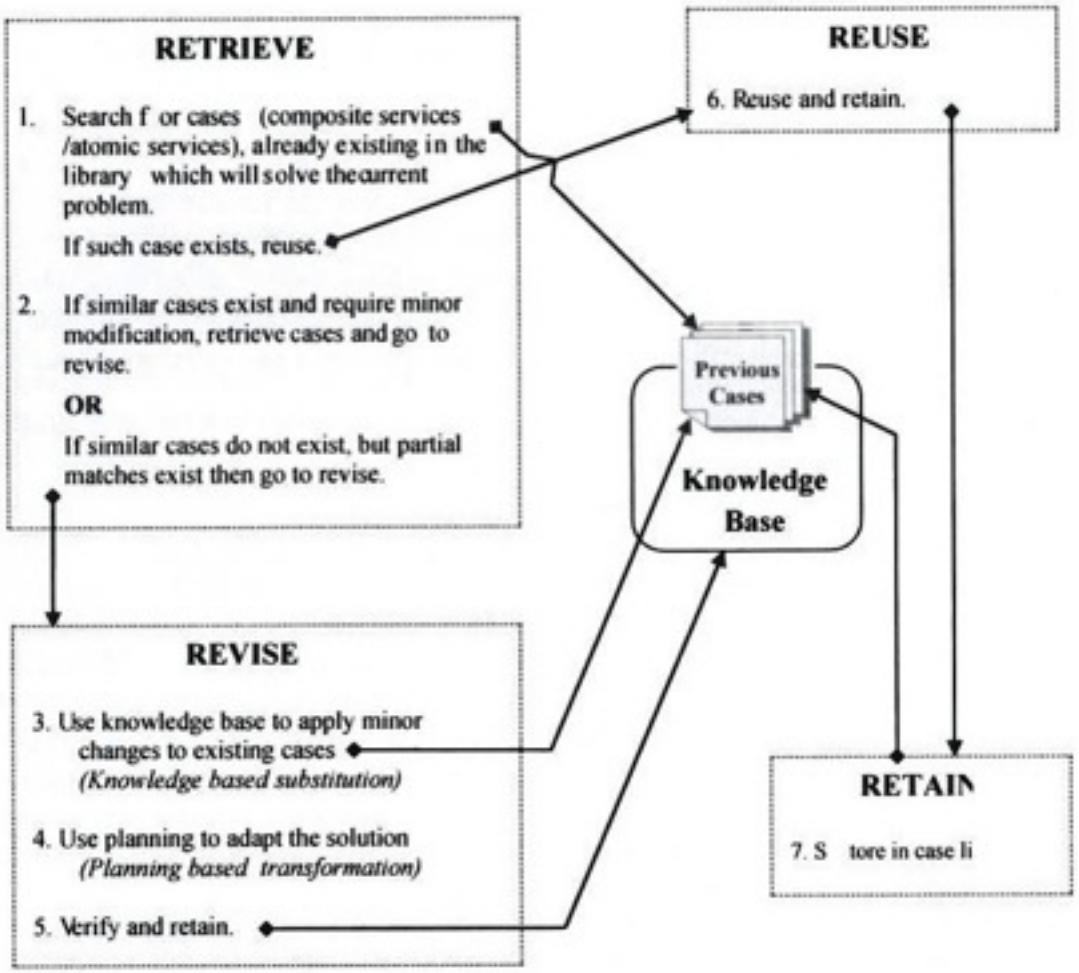
ρ indicates the application of planner for transformation, where classical planner handles the task of coming up with a sequence of actions that will achieve a goal (Russell & Norvig, 2003). The planning phase is a resource-intensive and computational expensive (Selman, 2000; Long & Fox, 2002) yet inevitable option, hence the two previous phases should narrow down the number of possible services planner can successfully use to generate a composed service.

Figure 8 on the shows the holistic CBR methodology to achieve Web services composition using the REVISE cycle.

RELATED WORK

Semantic descriptions are increasingly being used for exploring the automation features related to Web services discovery, matchmaking and composition. In (Zhang et al., 2003) such semantic-based approach is described. They use ontology to describe Web services templates and select Web services for composition by comparing the Web service output parameters with the input parameters of other available Web services. A constraint driven composition framework in (Aggarwal et al., 2004) also uses functional and data semantics with QoS specifications for selecting Web services. DARPA's OWL-S (Ontology Web Language for Web services) is the leading semantic composition

Figure 8. CBR methodology for Web services composition (modified from [(Aamodt et al., 1994)])



research effort. OWL-S (Martin et al., 2004b; Ankolekar et al., 2001) 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, Hendler, & Parisa, 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 techniques to build a task list to achieve composition objectives: selection of services and flow management for performing composition of services to match user preferences. (McIlraith & Son, 2002) uses Golog – AI planning Reasoner for automatic composition, while in a similar spirit some other approaches (Wu, Parisa, Hendler, Nau, & Sirin, 2006; Nau, Cao, & Lotem, 1999) have used the paradigm of Hierarchical Task Network (HTN) planning to perform automated Web service composition. These approaches use semantics for automatic Web services discovery, but they overlook the Web service execution behaviour in the decision-making process.

Use of CBR, Semantic Web and Web services are common technologies in our effort and the efforts in (Nern et al., 2006) with different objectives, their's being to consume these technologies to assist the procedure of Semantic Web services creation using Case-Based reasoning

approach, while our main concern is services composition.. Their INFRAWEBS project has Semantic Web Unit (SWU) – a collaboration platform and interoperable middleware for ontology-based handling and maintaining of Semantic Web services. The framework provides knowledge about a specific domain and relies on ontologies to structure and exchange this knowledge to Semantic Web services development process.

There is also a number of existing approaches which applies CBR for workflow modelling. (Madhusudan, Zhao, & Marshall, 2004) proposes an approach to support workflow modelling and design by adapting workflow cases from a repository of process models where workflow schemas are represented as cases and are stored in case repositories. The cases are retrieved for a problem which requires similar business process to solve the problem. The description and implementation language of framework is based on XML and main focus is on assisting workflow designer in creating business process flows. In similar line, (Cardoso & Sheth, 2005) represents adaptive workflow management system based on CBR and targets highly adaptive systems that can react themselves to different business and organization settings. The adaptation is achieved through the CBR based exception handling, where the CBR system is used to derive an acceptable exception handler. The system has the ability to adapt itself over time, based on knowledge acquired about past execution experiences that will help solve new problems. Our approach concentrates on Web services as a unit of computation to take advantage of highly accessible and loosely coupled nature of Web services technologies. We focus on utilising service execution experiences to best serve user requirements and encode the framework with semantics.

Experience based learning using CBR is a relatively old branch of Artificial Intelligence and Cognitive Science and is being used (Hammond, 1986; Ashley & Rissland, 1988) 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 was initially documented in (Limthanmaphon & 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. Similar approach described in (Diaz, Salgado, Moreno, & Ortiz, 2006) proposes an extension of UDDI model for web services discovery using category-exemplar type of CBR, where web services are categorized in domains and stored as exemplar (Porter & Bareiss, 1986) of particular domain. Their implementation of CBR reasoner facilitates UDDI registry by indexing the cases based on the functional characteristics of Web services. However, the approach does not take into consideration the importance of non-functional parameters in service selection and the use of semantics at CBR level is peripheral as they primarily use the UDDI based component for service discovery. UDDI is text-based leaving little scope for automation. 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.

CONCLUSION

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 article, 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 to execution and can only be generalized if such execution values are stored and reasoned for deciding service capability. AI planning and Intelligent Agent based reasoning methods offer rule-based reasoning methodology rather than experience-based. Hence, we used Case Based Reasoning method that allows capturing experiences and reasoning based on them.

We implemented a 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 both for describing the problem parameters and for implementing components of the CBR system: representation, indexing, storage, matching and retrieval. Our approach for modelling CBR as ontology-based reasoner achieves developer transparency and makes the framework 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 specialise in servicing a particular domain, hence interoperation can facilitate inter-domain orchestration. We believe that in this work we took a small step towards standardization at the reasoner level by describing the CBR reasoning model semantically

In this article we also presented the preliminary experimental results of our framework, which informally proved the correctness of our approach despite the relatively slow response time of the matchmaking process. The latter is primarily attributed to exporting external ontologies, which can be countered by utilising off-line caching of public ontologies. The experimental results also demonstrated the advantages of classifying user groups into profiles that have standard set of constraint rankings.

The final contribution of the article was documenting our investigation into extending the discovery and matchmaking algorithm to cater for web services composition. We discuss how we envisage exploiting the REVISE stage of the CBR cycle, i.e. case adaptation, to facilitate service composition. The article advocates an exhaustive *knowledge-based substitution* approach to adapt the functional and non-functional attributes of the candidate case to the requested solution before suggesting more complex and computationally taxing AI-based planning-based transformations that integrate the service profile of a number of cases to deliver candidate solutions.

The next stage of this research will involve the formal validation and implementation of our adaptation-based composition model.

REFERENCES

Aamodt, A. & Plaza, E. (1994). Case-based reasoning: foundational issues, methodological variations, and system approaches. *AI Communications, 7*, 39-59.

Aggarwal, R., Verma, K., Miller, J., & Milnor, W. (2004). Constraint Driven Web Service Composition in METEOR-S. In *2004 IEEE International Conference on Services Computing (SCC 2004)* (pp. 23-30).

Akkiraju, R., Farrell, J., Miller, J., Nagarajan, M., Schmidt, M.-T., Sheth, A. et al. (2005). Web Service Semantics - WSDL-S. In *A joint UGA-IBM Technical Note*.

Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F. et al. (2003). Business Process Execution Language for Web Services version 1.0. http://www-128.ibm.com/developerworks/library/specification/ws-bpel/ [On-line].

Ankolekar, A., Burstein, M., Lassila, O., Martin, D., McIlraith, S., Narayanan, S. et al. (2001). DAML-S: Semantic Markup for Web Services. In *International Semantic Web Working Symposium (SWWS)* (pp. 411-430).

Ashley, K. D. & Rissland, E. L. (1988). A case-based approach to modelling legal expertise. *IEEE*

Expert: Intelligent Systems and Their Applications, 3, 70-77.

Cardoso, J. & Sheth, A. (2005). Adaptation and Workflow Management Systems. In *International Conference WWW/Internet 2005* (pp. 356-364). Lisbon, Portugal.

Diaz, O. G. F., Salgado, R. S., Moreno, I. S., & Ortiz, G. R. (2006). Searching and Selecting Web Services Using Case Based Reasoning. In *Computational Science and Its Applications (ICCSA 2006)* (pp. 50-57).

Elaine, R. & Kevin, K. (1992). *Artificial Intelligence*. McGraw-Hill.

Hammond, K. J. (1986). Learning to anticipate and avoid planning problems through the explanation of failures. In *AAAI-86 conference* (pp. 556-560). Cambridge, MA: AAAI press/MIT press.

HP-JENA (2003). Jena- A semantic Web Framework for Java. http://jena.sourceforge.net/ [On-line]. Available: http://jena.sourceforge.net/

Kolodner, J. & Simpson, R. (1989). The MEDIATOR: Analysis of an early case based problem-solver. *Cognitive Science, 13,* 507-549.

Limthanmaphon, B. & Zhang, Y. (2003). Web service composition with case-based reasoning. In *Fourteenth Australasian database conference on Database technologies* (pp. 201-208). Adelaide, Australia.

Long, D. & Fox, M. (2002). Progress in AI Planning Research and Applications. *Upgrade/Novatica, 3,* 10-25.

Madhusudan, T., Zhao, L. J., & Marshall, B. (2004). A case-based reasoning framework for workflow model management. *Data & Knowledge Engineering archive, 50,* 87-115.

Maher, M. L. & Garza, A. G. d. S. (1997). Case-Based Reasoning in Design. *IEEE Expert: Intelligent Systems and Their Applications, 12,* 34-41.

Martin, D., PaolucciSycara, M., McIlraith, S., Burstein, M., McDermott, D., McGuinness, D. et al. (2004a). Bringing Semantics to Web Services: The OWL-S Approach. In *First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004)* San Diego, California, USA..

Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S. et al. (2004b). Semantic Markup for Web Services. http://www.w3.org/Submission/OWL-S/ [On-line].

McIlraith, S. & Son, T. C. (2002). Adapting Golog for Composition of Semantic Web Services. In *the Eighth International Conference on Knowledge Representation and Reasoning (KR2002)* (pp. 482-493).

Nau, D., Cao, Y., & Lotem, A. (1999). SHOP: Simple Hierarchical Ordered Planner. In *International Joint Conference on Artificial Intelligence (IJCAI-99)* (pp. 968-973).

Nern, J., Agre, G., Atanasova, T., Marinova, Z., Micsik, A., László, K. et al. (2006). INFRAWEBS Semantic Web Service Development on the Base of Knowledge Management Layer. *International Journal "Information Theories and Applications, 13,* 161-168.

Osman, T., Thakker, D., & Al-Dabass, D. (2005). Bridging the Gap between Workflow and Semantic-based Web services Composition. In *the 2005 IEEE/WIC/ACM International Joint Conference* (pp. 13-23). Compiègne, France.

Osman, T., Thakker, D., & Al-Dabass, D., Lazer, D., & Deleplanque, G. (2006). Semantic-Driven Matchmaking of Web services using Case-Based Reasoning. Accepted for the 2006 IEEE International Conference on Web Services (ICWS 2006), held on September 18-22, 2006, Chicago, USA.

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

PORTAL (2003). Portal Ontology. http://www.aktors.org/ontology/portal [On-line].

Porter, B. W. & Bareiss, R. E. (1986). PROTOS: An experiment in knowledge acquisition for heuristic classification tasks. In *First International Meeting on Advnces in Learning (IMAL)* (pp. 159-174). Les Arcs, France.

Remind (1992). ReMind Developer's Reference Manual. *Cognitive systems.*

Russell, S. & Norvig, P. (2003). *Artificial Intelligence: A Modern Approach.* (2 ed.) Prentice Hall, Pearson Education, Inc, Upper Saddle River, New Jersey.

Selman, B. (2000). Compute-Intensive Methods for Artificial Intelligence. *Annals of Mathematics and Artificial Intelligence, 28,* 35-38.

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

Watson, I. (1997). *Applying Case-Based Reasoning: Techniques for Enterprise Systems.* (1st ed.) Morgan Kaufmann Publishers.

Watson, I. & Marir, F. (1994). Case-Based Reasoning: A Review. *The Knowledge Engineering Review, 9(4), 1994, 9,* 355-381.

Wu, D., Parisa, B., Hendler, J., Nau, D., & Sirin, E. (2006). HTN Planning for Web Service Composition Using SHOP2. *Journal of Web Semantics, 1,* 1-30.

Zhang, R., Arpinar, B. I., & Aleman-Meza, B. (2003). Automatic Composition of Semantic Web Services. In *International Conference on Web Services* (pp. 38-41). Las Vegas, Nevada, USA.

Taha Osman is a senior lecturer and leader of the Semantic Web Services Research Network in the School of Computing and Informatics, Nottingham Trent University. He received a BSc honours degree in computing from Donetsk Polytechnic Institute, Ukraine in 1992. He joined the Nottingham Trent University in 1993 where he received an MSc in real-time systems in 1994 and was awarded a PhD in 1998 for his work on developing fault-tolerant distributed computing systems. He is an IEEE member and reviewer for the Computer Journal. *His research interests include the fault-tolerance of distributed systems, intelligent agent systems, and Semantic Web services.*

Dhavalkumar Thakker is working as an associate with Nottingham Trent University, Nottingham. He finished his PhD thesis in May 2008 in the area of Web services composition at the Semantic Web Services Research Network, Nottingham Trent University. His thesis advisors were Taha Osman and David Al-Dabass. Prior to joining PhD degree, he completed his masters degree in data communication systems from the Brunel University, London and bachelors degree from the Gujarat University, India. His current research interests are in the areas of Semantic Web based multimedia retrieval and Web services.

David Al-Dabass is emeritus professor and holds the personal chair of Intelligent Systems in the School of Science & Technology, Nottingham Trent University. He is a graduate of Imperial College, holds a PhD and has held post-doctoral and advanced research fellowships at the Control Systems Centre, UMIST, Manchester University. He is Fellow of the IET, IMA and BCS. He is founder and editor-in-chief of the International Journal of Simulation: Systems, Science and Technology*; and currently serves as chairman of the UK Simulation Society and served on the European Council for Modelling and Simulation. He has authored or co-authored over 170 scientific publications in intelligent systems modelling and simulation.*