Semantic-Driven Matchmaking of Web Services Using Case-Based Reasoning

Taha Osman, Dhavalkumar Thakker, David Al-Dabass
School of Computing and Informatics, Nottingham Trent University, Nottingham, UK
(Taha.Osman, Dhavalkumar.Thakker, David.Al-Dabass)@ntu.ac.uk
Daniel Läzer, Ghislain Deleplanque
Naval Academy of France, Ecole Navale, BP 600, 29240 Brest, France

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 paper, we present a novel approach that utilizes the case based reasoning methodology for modelling dynamic Web service discovery and matchmaking. Our framework considers Web services execution experiences in the decision making process and is highly adaptable to the service requester constraints. The framework also utilises OWL semantic descriptions extensively for implementing both the components of the CBR engine and the matchmaking profile of the Web services.

1. 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 [1][2], and non-functional parameters such as domain-specific constraints and user preferences [3].

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 paper begins with describing the motivation behind the work. In section 3, we overview the theory of CBR. Section 4 explains how we model Web services matchmaking using CBR. In section 5 we discuss the design of our matchmaking algorithm. The implementation of the framework and analysis of results are described in section 6 and 7. In section 8 we review related work and we present our conclusions in section 9.

2. 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 [4], 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 [5]. 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 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.

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 [6], 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.

3. 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 [7]. The processes involved in CBR can be represented by a schematic cycle comprising four phases [8]:

- 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:

i. Case representation

A case is a contextualised piece of knowledge representing an experience [8]. 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 to answer to the problem but also criticises of 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.

ii. Case storage & 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. These indices express information about the content of the case.

iii. 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 with the aim of finding the cases which are contextually similar to the new problem. The retrieval is done according to the index of the cases.

vi. Matchmaking

Matchmaking performs the comparison between these similar cases and the new request to verify if the possible solution is the one applied to the prior cases. 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 newly input case. It assigns importance ranking to properties of cases and then computes the degree of matching by comparing the cases for these properties [8]. The matchmaking process is thus performed on each retrieved case, and the most similar case of the input case is the one with the highest result. 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 attempt 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

4. Matchmaking Web services using Case Based Reasoning

4.1 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 and is similar in role with Web service requestor. Figure 1 illustrates a schematic diagram for our framework.

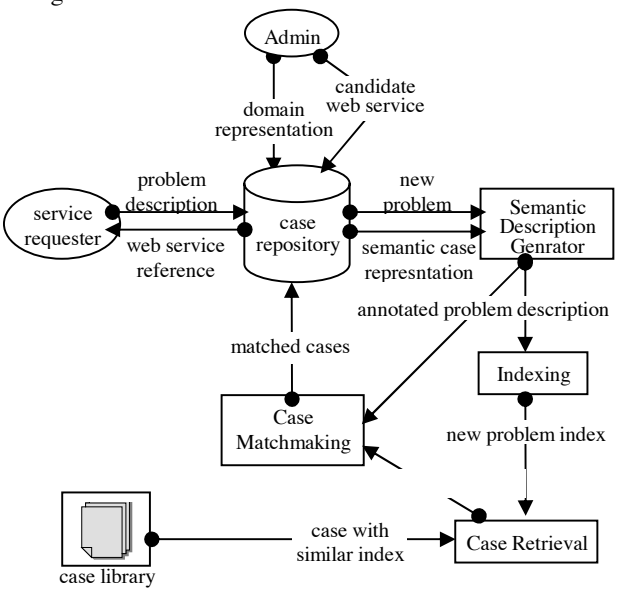


Figure 1. Architecture of the CBR matchmaking framework

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 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.

3. The case representation repository retrieves the appropriate semantic case representation format for the requested service and forwards it together with the problem description to the Semantic Description generator module, which semantically annotates the new problem according to the representation format.

4. 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.

5. The case retrieval module queries the case library for cases with the similar indexes. Output at this stage will be the cases that have similar index to the current problem, which will be candidates for matchmaking.

6. 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.

4.2 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.

4.2.1 Case vocabulary

In CBR theory**, t**he 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:

1. 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).

2. 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.

4.2.2 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 [9] 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.

The frame representations are highly structured and modular which allows handling 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 [10], 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. 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

The slot Travel Request corresponds to the Input, i.e. all the travel details as for any 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).

4.2.3 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.

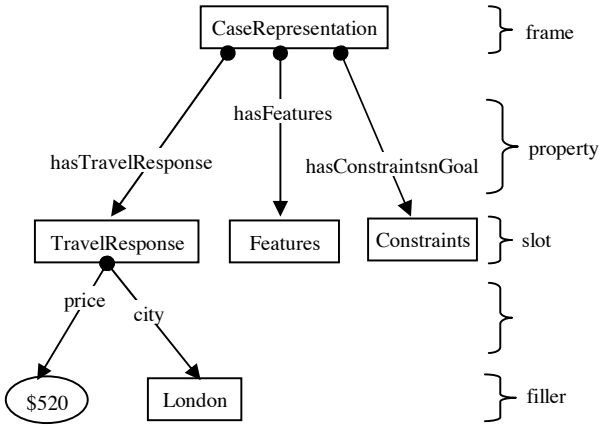


Figure 2. Mapping between frame structure and semantic case representation for travel domain

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**,** 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 [11], 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. He needs a quick result (approximately in 1.5 seconds)".

Table 2. Example of case

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

4.2.4 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).

5. Development of the CBR framework

5.1 Case indexing and Retrieval

To facilitate the search procedure, cases are indexed based on vocabularies. In our framework, we use "partitioning the case library" method, which is a variation of "flat memory indexing" technique [9]. 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 4) 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 efficency 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.

5.2 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 [12] method for our framework. The method operates as follows:

1. 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;
2. Quantify the weight of the similarity. A ranking is assigned to each property in accordance with its importance as exemplified in Table 3.

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

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*.

5.2.1 Data property comparison

To compare data type properties, like the price range or the value of QoS (e.g. execution time), we the qualitative regions based measurement method **[9].** 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: $|V_r - V_c| \leq X.[V_r|$, 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.

5.2.2 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 [13], 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 leaf 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:

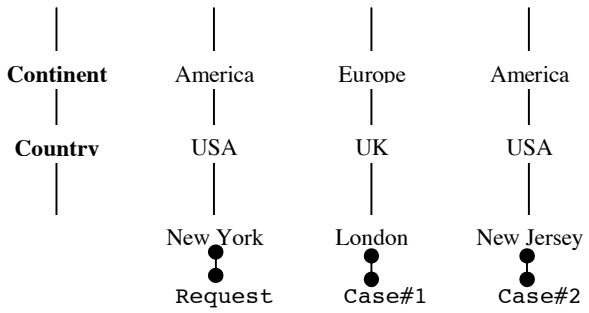
$$DoM = \frac{MN}{GN}$$

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.

Figure 3. Semantically matching object properties (dimensions)



5.2.3 Computing the overall similarity value

Overall similarity is evaluated by computing the aggregate degree of match (ADoM) [12] for each retrieved case according to the following equation:

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

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 step A, and takes aggregate of this sum by considering the importance of dimensions.

6. 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 [14] - a Java based OWL reasoner, as our ontology engine in favour of the more popular Jena [15], 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.

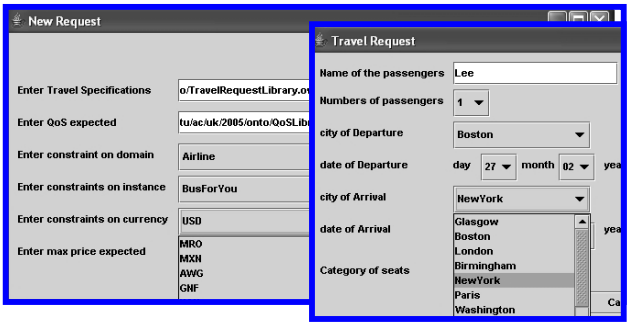


Figure 4. Admin and User Interface

7. 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 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.

Table 4. User Profiles

Profile	***Property***				
	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 8 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 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.

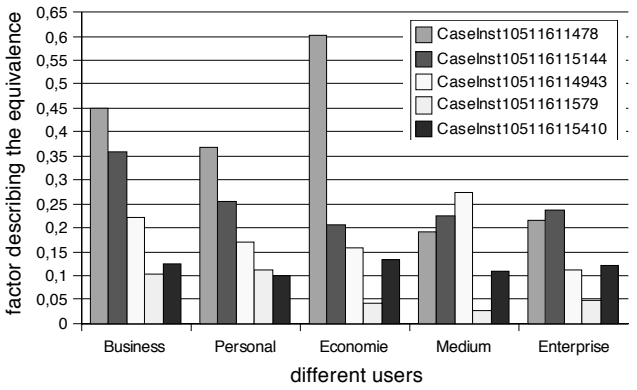


Figure 8. User Profiles

8. Related Work

Semantic descriptions are increasingly being used for exploring the automation features related to Web services discovery, matchmaking and composition. In [13] 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 [3] also 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 [16][17] 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 [18], 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.

9. Conclusions

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 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 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 the CBR system: representation, indexing, storage, matching and retrieval. These components are modelled based on ontologies, making the application logic captured within semantic descriptions. Our approach for modelling CBR as ontology-based reasoner achieves required automation 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 paper 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.

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. Adaptation is similar to Web service composition, as the composition is applied when available services are not sufficient in meeting the requirement for the problem.

10. References

[1] Martin, D., Paolucci, M., McIIraith, S., Burstein, M., McDermott, D ., McGuinness, D., Parsia, B., Payne, T., Sabou, M., Solanki, M., Srinivasan, N., and Sycara, K., : Bringing Semantics to Web Services: The OWL-S Approach, Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004).

[2] Akkiraju, R., Farrell, J., Miller, J., Nagarajan, M., Schmidt, M., Sheth, A., and Verma, K., Web Service Semantics - WSDL-S, A joint UGA-IBM Technical Note, version 1.0, April 18, 2005.

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

[4] Osman, T., Thakker, D., and Al-Dabass, D.,: 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, the 2005 IEEE/WIC/ACM International Joint Conference, Compiègne, France, (2005) 13-23.

[5] Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., and Weerawarana, S., "Business Process Execution Language for Web Services, Version 1.1", http://www-128.ibm.com/developerworks/library/wsbpel/

[6] Aamodt , A., and Plaza, E.,: Case-based reasoning: foundational issues, methodological variations, and system approaches, AI Communications, v.7 n.1, 1994 (39-59)

[7] WATSON, Ian. Applying Case-Based Reasoning : Techniques for Enterprise Systems.
San Francisco : Morgan Kaufmann Publishers. 1997. 285 p. ISBN 1-55860-462-6

[8] AAMODT, Agnar ; PLAZA, Enric. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AI Communications. IOS Press, Vol. 7:1. 1994. pp 39-59.

[9] Kolodner., J., and Simpson, R., The MEDIATOR: Analysis of an early case based problem-solver. Cognitive Science, 13(4): (1989) 507-549

[10] Rich, E., and Knight, K., Artificial Intelligence. McGraw-Hill, 1992.

[11] Portal Ontology. AKT Technologies. 10 Feb 2003. Available at: http://www.aktors.org/ontology/portal#

[12] ReMind Developer's Reference Manual, Cognitive Systems, Boston, 1992

[13] Zhang, R., Budak I., and Aleman-Meza, B.: Automatic Composition of Semantic Web Services. Proceedings of the International Conference on Web Services, ICWS '03, Las Vegas, Nevada, USA. (2003) 38-41

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

[15] Jena- A semantic Web Framework for Java, HP Labs Semantic Web Programme, http://jena.sourceforge.net/

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

[17] Ashley, K.D., and Rissland, E.L.,: A case-based approach to modelling legal expertise. IEEE Experts 3(3), (1988) 70-77.

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