How to apply CBR methods in Web service composition?

Soufiene Lajmi[1], Chirine Ghedira[2], and Khaled Ghedira[1]

[1] SOIE/ University of Tunis, Tunisia
Soufiene.Lajmi@ensi.rnu.tn
Khaled.Ghedira@isg.rnu.tn
http://www.soie.isg.rnu.tn
[2] Claude Bernard Lyon 1 University, France
Chirine.Ghedira@liris.cnrs.fr
http://www.liris.cnrs.fr

Abstract. The emergence of web services as a new technology supporting the future Web has motivated several researchers to investigate in this field. In addition, the possibility of selecting and integrating Web services, in or between organizations, is very useful for the improvement of their profitabilities. In such a case, we introduce the term of *web service composition*. In this paper, we propose an approach called WeSCo_CBR (Web Service Composition founded on Case Based Reasoning) that aims at enhancing the process of web service composition by using case based reasoning technique which originates from artificial intelligence. Furthermore, we regard efficient description and management of information semantics as a major requirement to enable semantic interoperability. We opt to integrate ontology so that we can apply a reasoning to help perform meaningful web service composition.

Key words: Web Services (WS), Web Service Composition, Case-Based Reasoning (CBR), Ontology

1 Introduction

With the proliferation of Web services in the last years, distributed architectures have seen a real development. Indeed, not only does WS ensure the interoperability between applications which are supported by heterogeneous systems, but also their composition appears as an important strategy to provide more complex and useful solutions [16].

Composition addresses the situation of a request that cannot be answered by existing and/or elementary WS, whilst it has favored the re-use and the creation of services. However, the simple use of the existing WS standards (such as Simple Object Access Protocol SOAP [1], Web Service Description Language WSDL [15] and Universal Description, Discovery and Integration protocol UDDI [3]) does not ensure a dynamic and efficient composition. In [20], authors have discussed two approaches: industry approach and semantic web approach. For the first one, many research issues, considering process description specifications, have been proposed such as WSFL [5], Xlang [6], BPEL4WS [7]. However, none of these suggested specifications actually treat the dynamic business process creation. Indeed, one of the requirements imposed by these specifications is that the process must be pre-defined. For the semantic web solution, several initiatives have proposed languages and frameworks such as Web Services Modelling Ontology WSMO [14], WSDL-S [17] which aim at integrating a semantic level to describe web services in order to improve the WS composition process or OWL-S [13] language which has defined *Service class* to model WS with the properties *presents*, *describedBy* and *supports*. OWL-S ontology is divided into three sub-ontologies: the *Service Profile* to describe what the service can do, the *ServiceModel* to describe the control flow and dataflow and the *ServiceGrounding* to specify the details of how a service can be acceded.

Despite these efforts, the WS composition is still a complex task and is beyond the human ability to make the composition plan manually. Semi-automated/fully dynamic web services composition hence presents yet a real challenge.

A crucial step in the process of developing applications based on WS is the WS discovery. It consists in finding the WS that fulfils better a user request. The main obstacle affecting this step is the huge number of the available WS. A study on the WS discovery mecanisms has been done in [18]. In our work, for the WS discovery, we are only interested in reducing the search space of WS. Thus, the first WS in the space will be selected.

In this paper, we present the WeSCo_CBR approach and demonstrate how CBR techniques are applied for semi-automated WS composition. Our contribution consists of: 1) the definition of an abstract level defined by a set of domain activities in order to reduce the search space of WS, 2) a method to build an abstract process (through a set of similarity computation procedures according to the CBR techniques) and then a composite WS.

The rest of this paper is organized as follows: section 2 motivates the adoption of the ontology in conjunction with the CBR technique, section 3 deals with the foundations and principles of our approach, section 4 presents the implementation of the system, section 5 discusses related work, and finally, we present our conclusions and discuss future research in section 6.

2 Background

2.1 Motivating Scenario

Our scenario concerns the medical field. To illustrate the goals of our proposal, we consider a request for a medical diagnosis of the early detection of cardiac ischemia and arrhythmia. This request consists in carrying out a cardiologic diagnosis of a patient, starting from the analysis of his electrocardiogram (ECG). In fact, a patient has a Portable ECG Monitor (PEM), which is used to detect and manage any cardiac event. When the patient feels a chest pain, he turns on the PEM so his ECG is recorded. The PEM starts with a serial analysis of this record and compares it with the referenced ECG. The PEM service can suspect any cardiac problems and should look for a call center service to send an alert, if needed. The alert triggers a WS whose role is to find a first-aid medical center close to the patient's current location. Processing both the recorded and referenced ECG, the selected medical center identifies the type of alert: severe or minor.

Without the use of the abstract activities, the response to this request requires its comparison with the whole content of the repository of the actual WS (even those which do not have any relationship with the medical diagnosis). This explains the difficulty and the high cost of discovery and selection tasks of the actual WS which covers this request.

By abstract activity, we mean a concept, described in an ontology, which presents a set of real services having the same functionality. Thus, the fact of using an abstract process (a set of ordered abstract activities) facilitates the discovery and selection tasks by reducing the scope of search. Whenever an abstract process is required, it is sufficient to start, for each activity, a search for the WS represented by this activity.

2.2 Why using the Case-Based Reasoning?

Our adoption of the Case-based reasoning is supported by various reasons. First, Case-based reasoning [11, 19] is a problem solving paradigm which, in many respects, is fundamentally different from other major AI approaches. Indeed, CBR is the process of solving new problems based on the solutions of similar past problems. In other terms, instead of relying solely on general knowledge of a problem domain, or making associations along generalized relationships between problem descriptors and conclusions, CBR is able to utilize the specific knowledge of previously experienced, concrete problem cases. A new problem is solved by finding a similar past case, and reusing it in the new problem case. Second, CBR is also an approach to incremental, sustained learning, since a new experience is carried out each time a problem has been solved, making it immediately available for future problem solving. Finally, it has been argued that CBR is not only a powerful method for computer reasoning, but also a pervasive behavior in every day human problem solving.

In our study, we propose to apply the CBR method, combined with the use of OWL-S as a language to describe and develop the abstract processes. This type of reasoning consists in finding, in the case base, cases similar to a new user request. In addition, we estimate that the use of ontology is very important for the achievement of a composition platform. Indeed, ontology provides a rich description of the resources which allows to improve the search for the most relevant services and their selections. Moreover, the OWL-S enables the automatically discovery, execution, composition, and interoperability WS [20]. Once the relevant services are selected, the last stage is the construction of a composite web service. In the next section, we present the architecture of WeSCO_CBR.

3 A proposal for web service composition based on CBR

In this section, we briefly present the architecture of WeSCo_CBR, next we will proceed with the description of the use of CBR for the semi-automated WS composition.

3.1 WeSCO_CBR

In [4], we `have presented` the architecture of our proposal which consists of five components: 1) the *Activity Discovery Engine* which allows to find the set of activities that best fits a user request, 2) the *Abstract Process Binder* whose role is to build an abstract process made up of a set of abstract activities, 3) the *Web Service Discovery Engine* which allows to determine-for each activity of this process-the integrality of the actual semantic WS which can substitute the activity, 4) the *Selector* which includes the selection mechanism[3] of one of these discovered semantic WS. This enables to make a comparison between the requested activity and the integrality of the semantic WS proposed by the *WS Discovery Engine*, 5) the *Constructor* which allows the transformation of the abstract process built by the *Abstract Process Binder* in an executable process. It is the component which generates an executable OWL-S process that can be executed by the dedicated engine.

3.2 How to apply CBR in WeSCo_CBR?

In order to simplify the request processing, we need to transform the user request in an easy-to-handle language, understandable by the machine. The reformulation step translates the request in an easy ontological formula. This task is launched by the *Activity Discovery Engine*. To do so, we propose to divide the request into three components defined as follows:

– Instances: A request can contain a set of data. These data can be considered as values for attributes of one or more objects. The *Instances* part of the request represents the classes inherited by these objects.
– Variables: A request can contain variables. These variables can be considered as attributes of one or more classes which represent the *Variables* part of a request.
– Activities: It is the set of abstract activities of a request. Indeed, the activities of a request are deduced from variable and instance classes.

Once the *Instances* and *Variables* parts of the request are set, a search for activities is launched by the *Activity Discovery Engine*. This engine allows us to obtain all the activities which fit in the request. The activity search algorithm shown below is based on the ontological description of the activities for the medical domain classes. *Activity search algorithm*

[3] Until now, the first web service among the selected ones will be selected

```
function CActivityList RequestActivities(CVariableList,
                                         CInstanceList)
Input:   CVariableList {List of Variable classes}
         CInstanceList {List of Instance classes}
Output:  CActivityList {List of Activities}
Begin
   CActivityList result=new CActivityList()
   CVar {Variable class}, CIns {Instance class}
   for CVar in CVariableList do
        result.add(CVar.ActivityList())
   end for
   for CIns in CInstanceList do
        result.add(CIns.ActivityList())
   end for
   return(result)
End
```

For the needs of our study, we have defined a local ontology. It is described in Fig. 1 and presents some concepts and activities and further relationships between them. The *ActivityList* function shown below enables-for each concept-to find activities that have relation with it.

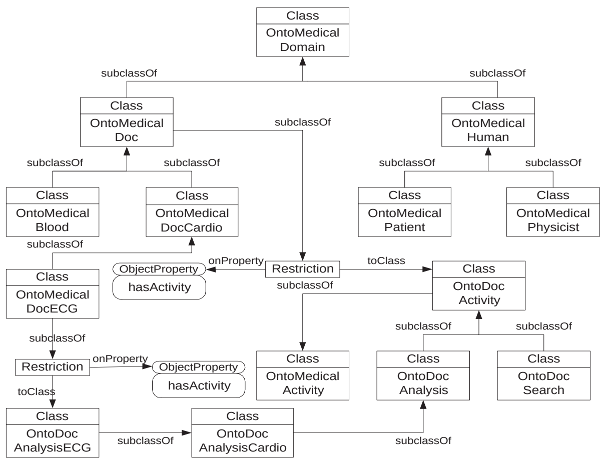

Fig. 1. Ontology illustration of the description of the activities and concepts in the medical domain

ActivityList algorithm

```
function CActivity[] ActivityList( )
Begin
```

```
    OntModel ontology=owlModel.getOntModel();
    RDFProperty HasActivity=owlModel.getRDFProperty("hasActivity");
    String queryString = "SELECT ?domain, ?range\n"
            +"WHERE (?domain  onto:hasActivity  ?range)\n"
            +"AND ?domain=~/"+CV+"/"
            +"USING onto FOR <http://127.0.0.1/MedicalField.owl#>" ;
    Query query = new Query(queryString) ;
    query.setSource(ontology);
    QueryExecution qe = new QueryEngine(query) ;
    Individual Ind;
    QueryResults results = qe.exec() ;
        for ( Iterator iter = results ; iter.hasNext() ; )
        {
                ResultBinding res = (ResultBinding)iter.next() ;
                Object range= res.get("range") ;
                Object domain =res.get("domain") ;
                Ind=ontology.getIndividual(range.toString());
                Activities.add(Ind.getLocalName());
        }
        results.close() ;
        return(Activities) ;
End
```

As mentioned before, our approach is based on CBR to construct the abstract process made up of the request activities. We believe that an intelligent method can provide better solutions. The CBR technique uses existing cases to provide a process to a user's request. In the first step, the proposed process is the solution of the more similar case to the request. This solution may not use some required services. To do so, we build-in the second step-another request composed of those services. Finally, we obtain two processes. The first process will be adapted with reference to the second. However, the use of CBR requires the identification of a case, which needs to be represented by a model adapted to our problematic. This modelling allows us to describe each component of a case. For the search of similar cases and the selection of the relevant cases, we need the case similarity computation and search procedures.

In the first part of this section, we define the representation of a case. In the second part, we present the methods elaborated to calculate the similarities.

Case representation. According to Kolodner[11], regardless of the applicability domain, a case has always the same components. These components are a t-uple composed of a problem, a solution and possibly an evaluation. Likewise, in WeSCo_CBR [4], a case is composed of the following three elements: 1) *the problem* which is composed of four parts: user's profile, activities, variable classes and instance classes, 2) *the solution* which is the combination of a set of activities, 3) *the evaluation* which is the relevance ratio of the solution. Due to the existence of irrelevant cases which does not fit the user's needs, we propose evaluation criteria of the user to express his satisfaction degree to the suggested process.

After the elaboration of the case modelling, it is necessary to establish the discovery and selection procedures for the most relevant cases. In the following, we deal with the problems of the search and selection of a case for a new request presented by the user.

Similar case search system. For a new request, the re-use process consists in looking for a memorized similar case and, if required, in evaluating and memorizing the new case. Moreover, we need to find, for each request, the most relevant memorized case which can best fit this request. This process is composed of the following stages: 1) *Representation of the problem*: for each new request, we search the most relevant cases. This research is carried out according to the request (problem). To do so, we express the request in the form of a new case in order to be able to compare it with memorized former cases. This stage has been processed in section 3.2, the following stages will be analyzed in this section. 2) *Similarity computation*: the most relevant case is generally given according to its similarity with the new case. With this intention, we define methods for similarity computation in order to guide the research. Accordingly, we propose some methods to calculate the similarity between cases. Similarity computation is done for the components (problem part) of the request. 3) *Procedure of research of the most relevant case*: it uses the methods of similarity computation to propose the most relevant case evaluated by procedures of search and similarity described in this section. 4) *Memorizing*: action left for the user to judge if the new case is interesting enough to memorize. In the same way, we propose to the application user the choice to memorize its new case. The memorized case is followed by an evaluation which will make it possible to refine the following research. In [4], we have presented similarity computation between activities. It is based on two algorithms. The first one allows to calculate the similarity between two activities. The second concerns the case similarity computation in terms of activities. In the next section, we present the similarity computation between variable classes.

Similarity computation between variable classes. The variable classes represent the second dimension for the calculation of similarity between a new case and a memorized case. The similarity between two cases according to their variable classes is expressed by the following formula:

$$Sim_v(NC, MC) = \frac{\sum_{i=1}^{NV} Sim_v(NCV_i, MC)}{\max(NV, MV)} \tag{1}$$

where

- NC is a new case;
- MC is a memorized case;
- NV and MV are, respectively, the number of NC variable classes and those of MC variable classes;

– NCV_i is a variable class of NC.

The similarity between a variable class of a new case and all variable classes of a memorized case is equal to the maximum of the similarities between this class with each variable class of the memorized case. The following formula presents calculation of this similarity.

$$Sim_v(NCV_i, MC) = \max_{j=1..MV}(Sim_v(NCV_i, MCV_j)) \tag{2}$$

where MCV_j is a class of variable of MC.

The similarity between two variable classes is expressed by the following formula:

$$Sim_v(MCV_i, MCV_j) = \begin{cases} 1 & \text{if } MCV_i = MCV_j, \\ x^d & \text{if } MCV_i \text{ is a subclass} \\ & \text{of } MCV_j, \\ x^{2d} & \text{if } MCV_j \text{ is a subclass} \\ & \text{of } MCV_i, \\ x^{3d} & \text{else.} \end{cases} \tag{3}$$

d represents the maximum of the distances between NCV_i and MCV_j with their common parent.
x = (2* the number of common properties between NCV_i and MCV_j) / sum of the properties of NCV_i and MCV_j.

In the case base, we can have a very significant number of memorized cases. However, it can happen that none of these cases answers exactly the request. In this case, the case selected by the algorithms of search is a relevant case but it requires some processing and modifications.

4 Implementation

An overview of the implementation details is presented in this section. We then illustrate WeSCo_CBR by using a scenario from the medical domain. All components are implemented by jdk 1.5.0 using JBuilder 7.0. as a development environment. We use existing semantic web tools, like OWL, OWL-S, Protege 3.0 and Jena 2.1 to provide the basic representation and reasoning technology. Activities, Variable classes and Instance Classes are described using OWL whereas we use OWL-S to create semantic WS. To save the cases, we have created a MySQL database. The *Constructor* uses the OWL-S API to create a composite WS from an abstract process provided by the *Abstract Process Binder*.
To reason on the ontology, we chose to use Jena 2.1 which is a Java API for semantic web applications. This API deals with searching and reasoning on ontology such as the ones we have defined. To demonstrate the feasibility of our proposal, we have developed a graphical user interface that allows to enter a request. The user can launch the reformulation request process. An OWL file is generated to describe the different components of the reformulated request. This OWL file is then used by the *Abstract Process Binder* to generate the solution

of the user request.
We can show the solution proposed by our prototype. It can be modified manually by the user and adapted by creating another request for a new sub-problem.

5 Related Work

Recently time, several approaches to applying AI techniques to the web service composition problem have been published. Other approches are based on the workflow. EFlow [8] uses a static workflow generation method. A composite service is modelled by a graph that defines the order of execution. In Meteors [9], the authors have proposed an approach which consists in adding the semantic to the current standards such as UDDI, WSDL and BPEL. However, those two initiatives require a predefined workflow, which can be a major disadvantage. Some other work is based on the technique of artificial intelligence planning. SWORD [10] presents one of these works. However, SWORD does not use the emerging service description standards. [15] has presented an approach that supports running views over the specification of a composite web service. This approach is based on context and constraints to generate a derived state chart diagram[4] from an other state chart (initial or derived). In fact, if the constraints on an incoming transition of a service chart diagram is not satisfied in a certain context, then this service chart diagram will be excluded from the derived state chart diagram. However, a derived specification does not accept extra services through their service chart diagrams. In our proposal, we suggest to use an approach based primarily on the Case Based Reasoning (CBR) that has been dealt with in paragraph 3.2. So that, we can integrate existing cases to generate a composite web service which responds to the user request. However, the selection of the first discovered web service can provide an unsuitable solution. So, the selection mechanism must be improved.

6 Conclusion and future work

Recently, several technologies and languages interfering in various stages of WS life cycle have been developed. However, those single technologies do not allow an effective and dynamic composition of WS. For this purpose, WeSCo_CBR combines the CBR technique and the semantic description of Web services. It is characterized by the following advantages: firstly, the use of ontology and description semantics of the activities and the services allowing the best reasoning in the various steps of the composition. Secondly, it allows a dynamic composition of WS starting from a user's request. Our future works will focus primarily on the distribution of WeSCo_CBR using a multi-agent system. Indeed, we estimate that the distribution of the repository of activities, as well as the repository of the WS is very important. Therefore, we propose to use intelligent agents to decentralize the WS composition process.

[4] A state chart diagram is used as a means for modelling and specifying the component web services of a composite service

References

1. Moreau, J.J.: Introduction to soap awalkthrough of core technical concepts. XML EUROPE (2004)
2. Mantek, F.: What's new in wsdl. Wrox Conferences (2002)
3. Chauvet, J.M.: Services web avec soap, wsdl, uddi, ebxml. Paris: Eyrolles, Vol. 99:524p. Paris (2002)
4. Lajmi, S., Ghedira, C., Ghedira, K., Benslimane, D.: Wesco_cbr: How to compose web services via case based reasoning. The IEEE International Symposium on Service-Oriented Applications, Integration and Collaboration held with the IEEE International Conference on e-Business Engineering (ICEBE 2006), Shanghai, China (October 2006)
5. Leymann, F.: Web services flow language (wsfl 1.0). (May 2001)
6. Levy, D.: Coordination of web services : langages de description et plate-formes d'exécution. (Septembre 2002)
7. Juric, M., Sarang, P., Mathew, B.: Business process execution language for web services. page 270 (octobre 2004)
8. Casati, F., Ilnicki, S., Jin L.: Adaptive and dynamic service composition in eflow. In Proceedings of 12th International Conference on Advanced Information Systems Engineering(CAiSE), Stockholm, Sweden (June 2000)
9. Aggarwal, R., Verma, K., Sheth, A., Miller, J., Milnor, W.: Constraint driven web service composition in meteor-s. Submitted to 2004 IEEE International Conference on Services Computing (2004)
10. Ponnekanti, S.R.,Fox, A.: Sword: A developer toolkit for web service composition. In Proceedings of the 11th World Wide Web Conference, Honolulu, HI, USA (2002).
11. Kolodner, J.L.: Case-based reasoning. San Mateo, CA : Morgan Kaufman, (7), (1993).
12. Limthanmaphon, B., Zhang, Y.: Web service composition with case-based reasoning. In Proceedings of The 14th Australasian Database Conference (February 2003)
13. Burstein, M., Ankolenkar, A., Paolucci, M., Srinivasan, N.: Owl-s: Semantic markup for web services (2003)
14. Arroyo, S., Stollberg, M.: WSMO primer. SMO Delivrable D3.1, DERI Working Draft, Technical reportl (2004)
15. Benslimane, D., Maamar, Z., Ghedira, C.: A view based approach for tracking composite web services. ECOWS, Vxj, Sweden, IEEE Computer Society (2005)
16. Ghedira, C., Maamar, Z., Benslimane, D. On composing web services for coalition operations - concepts and operations-. International Journal Information & Security. Special issue on Architectures for Coalition Operations (2005) 16:79–92
17. Miller, J., Verma, K., Shelth, A., Aggarwal, R., Sivashanmugan, K.: WSDL-S: Adding semantics to wsdlwhite paper. Technical report, Large Scale Distributed Information Systems. (2004)
18. Garofalakis, J., Panagis, Y., Sakkopoulos, E., Tsakalidis, A.: Web Service Discovery Mechanisms: Looking for a Needle in a Haystack?. International Workshop on Web Engineering. (2004)
19. Leake, B.: CBR in Context: The Present and Future. In Leake, D., editor, Case-Based Reasoning: Experiences, Lessons, and Future Directions, AAAI Press/MIT Press. (1996)
20. Srivastava, B., Koehler, J.: Web Service Composition Current Solutions and Open Problems. Workshop on Planning for Web Services ICAPS (2003)