PreDiCtS: A Personalised Service Discovery and Composition Framework

Charlie Abela, Matthew Montebello

Department of Computer Science and AI
University of Malta
{charlie.abela, matthew.montebello}@um.edu.mt

Abstract. The proliferation of Web Services is fostering the need for applications to provide more personalisation during the service discovery and composition phases. An application has to cater for different types of users and seamlessly provide suitably understandable and refined replies. In this paper, we describe the motivating details behind PreDiCtS[1], a framework for personalised service discovery and composition. The underlying concept behind PreDiCtS is that, similar service composition problems could be tackled in a similar manner by reusing past composition best practices. These have to be useful and at the same time flexible enough to allow for adaptations to new problems. For this reason we are opting to use template-based composition information. PreDiCtS's retrieval and refinement technique is based on conversational case-based reasoning (CCBR) and makes use of a core OWL ontology called CCBROnto for case representations.

Keywords: CCBR, Ontologies, Semantic Web, Web services

1. Introduction

Reusability and interoperability are at the core of the Web Services paradigm. This technology promises seamlessly interoperable and reusable Web components that facilitate rapid application development and integration. When referring to composition, this is usually interpreted as the integration of a number of services into a new workflow or process. A number of compositional techniques have been researched ranging from both, manual and semi-automatic solutions through the use of graphical authoring tools [18], [19], to automated solutions based on techniques such as AI planning [17] [20] and others.

The problem with most of the composition techniques mentioned above is three fold (i) such approaches attempt to address service composition by composing web services from scratch, ignoring reuse or adaptation of existing compositions or parts of compositions, (ii) it is assumed that the requester knows exactly what he wants and

1 This research has partially been funded by the European Commission and by the Swiss Federal Office for Education and Science within the 6th Framework Programme project REWERSE number 506779

how to obtain it and (iii) composing web services by means of *concrete* service interfaces leads to tightly-coupled compositions in which each service involved in the chain is tied to a web service instance. Using this approach for service reuse, may lead to changes in the underlying workflow which range from slight modifications of the bindings to whole re-designing of parts of the workflow description. Therefore in our opinion, services should be interpreted at an abstract level to facilitate their independent composition. [10] adds, *"abstract workflows capture a layer of process description that abstracts away from the task and behaviour of concrete workflows"*, and this allows for more generalisation and a higher level of reusability. A system can start by considering such abstractly defined workflow knowledge and work towards a concrete binding with actual services that satisfy the workflow.

To make effective reuse of such abstract workflow definitions one could consider CBR, that is amenable for storing, reusing and adapting past experience for current problems. Nevertheless CBR restricts the user to define a complete problem definition at the start of the case-retrieval process. Therefore a mixed-initiative technique such as CCBR [3] is more appropriate since it allows for a partial definition of the problem by the user, and makes use of a refinement process to identify more clearly the user's problem state.

In summary we have identified the following motivating points:

1. Reusability of compositions has the advantage of not starting from scratch whenever a new functionality is required.
2. For effective reusability a higher level of abstraction has to be considered, which generalises service concepts and is not bound to specific service instances.
3. Personalisation of compositions can be achieved by first identifying more clearly the user's needs and then allowing for reuse and adaptation of past compositions based on these needs prior to binding with actual services.

The goal of this work is to present, the motivation behind, and prototype of PreDiCtS, a framework which allows for personalisation of service discovery and composition through the reuse of past composition knowledge. One could say that we are trying to encode and store common practices of compositions which could then be retrieved, reused and adapted through a personalisation technique. The solution we propose in PreDiCtS has two phases.

For the first phase, which we call the *Similarity Phase*, we have adopted a mixed-initiative technique based on CCBR. This provides for the personalisation process. Given a new problem or service composition request, this approach allows first to retrieve a ranked list of past, similar situations which are then ranked and suggested to the requester. Through a dialogue process the requester can decide when to stop this iterative-filtering phase, and whether to reuse or adapt a chosen case. Case definition is through an OWL-based ontology which we call CCBROnto [2] and which provides for the description of context, problem and solution knowledge At present PreDiCtS allows for case creation and retrieval (adaptation is in the pipeline) and once a case (or set of cases) is retrieved, it can be presented to the next phase, which we call the *Integration Phase* where a mapping is attempted, from the features found in the chosen solution, to actual services found in a service registry. Due to space restrictions this is dealt with in a future paper.

The rest of this paper is organized as follows. In Section 2 we will give some brief background information on CCBR. Then in Section 3 we will give an overview of the OWL case ontology, CCBROnto. In Section 4 we will present the architecture of PreDiCtS and some implementation details mainly focusing on the case-creator and case-retriever components. After which we present the last section with future work and concluding remarks.

2. Conversational Case-Based Reasoning

Case-Based Reasoning is an artificial intelligence technique that allows for the reuse of past experience to solve new problems. The CBR process requires the user to provide a well-defined problem description from the onset of the process. But users usually cannot define their problem clearly and accurately at this stage. On the other hand, CCBR allows for the problem state to be only partially defined at the start of the retrieval process. Eventually the process allows more detail about the user's needs to be captured by presenting a set of discriminative and ranked questions automatically. Depending on the user's supplied answers, cases are filtered out and incrementally the problem state is refined. With each stage of this problem refinement process, the system presents the most relevant solutions associated to the problem. In this way the user is kept in control of the direction that this problem analysis process is taking while at the same time she is presented with solutions that could solve the initial problem. If no exact solution exists, the most suitable one is presented and the user is allowed to adapt this to fit her new requirements. Nevertheless, this adaptation process necessitates considerable domain knowledge as explained in [4], and is best left for experts.

One issue with CCBR is the number of questions that the system presents to the user at every stage of the case retrieval process. This issue was tackled by [11] which defined question-answer pairs in a taxonomy and by [1] through the use of knowledge-intensive similarity metrics. In PreDiCtS we have adapted the former method[2] since a QA pairs taxonomy is defined to be an acyclic directed graph in which nodes are related to other nodes through parent-child relations and it is assumed that a node subsumes all its descendent nodes. This is very similar to how classes in OWL are related via the *subClassOf* relation and this fits well with the underlying case structure that we use in PreDiCtS.

3. CCBROnto

CCBROnto is an important component of PreDiCtS since it provides for (i) case and question-answer pair definitions, and (ii) the association of domain and case-specific knowledge. In CCBROnto the topmost concept is a *Case*. Its basic components are defined by the *CaseContext*, *Problem* and *Solution* classes. In [8] context is defined as "*any information that can be used to characterize the situation of an entity. An entity*

[2] Whenever we refer to this taxonomic theory we will be referring the work done by Gupta

is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves". We fully agree with this definition and in the *CaseContext*, we have included knowledge related to the case creator, case history, ranking and case provenance. We have considered ideas presented in [6], [7] and [15] which discuss the importance of context in relation to Web Services and stresses on the importance of the use of context in CBR, especially when cases require adaptation. Such context knowledge makes it possible to differentiate between users and thus the system could adapt cases accordingly. For example in the travelling domain, both going to a conference and going for a holiday may require similar services, such as hotel booking and flight reservation, though the use of a conference booking service is only required in the former. Thus, based on the contexts or roles of the users (a researcher the former and a tourist the latter) the CBR system can adapt the case knowledge to present cases that satisfy the requirements of both. A researcher can adapt the case for the tourist by including a suitable conference booking service.

In PreDiCtS we consider highly important such context knowledge since it helps to identify, why a case was created and by whom, together with certain aspects of case usage and its relevance to solving a particular problem. The *CaseCreator* provides a *Role* description that the creator associates himself with, together with a *foaf:Person* instance definition that describes who this person is. The motivation behind using foaf is to eventually be able to embed some level of reputation relevant to the person who created the case. The importance of this feature will become more visible and important when cases are shared.

The *CaseContext* also provides a place holder for *CaseHistory*. The knowledge associated with this feature is important when it comes to case ranking and usage, since it allows users to identify the relevance and usefulness of a case in solving a particular problem. It is also important for the case administrator when case maintenance is performed. Cases whose history indicates negative feedback may be removed from the case base. Case *Provenance* is also used in conjunction with reputation since it indicates a URL from where the case originated. Encapsulating such information in each case will help in maintaining a reliable case base.

The *Problem* state description in a PreDiCtS case is based on the taxonomic theory. Every problem is described by a list of QA pairs rather then a bag. This is required since QA pairs have to be ranked when they are presented to the user. Each *QAPair* is associated with a *CategoryName*, a *Question* and an *Answer* (see Fig.1). Each question has a textual description and is associated with a concept from the domain ontology through the *isRelatedTo* relation. We further assume that Answers could be either binary or nominal-valued. For this reason we have created two types of answer classes, *YesNoAnswer* and *ConceptAnswer*. The former is associated with a literal represented by either a *Yes* or a *No*. While the latter, requires an association with a concept in some domain ontology, through the previously mentioned *isRelatedTo* property. The motivation behind the use of this property is related to the taxonomic theory, which requires that QA pairs are defined in a taxonomy so that during case retrieval, the number of redundant questions presented to the requester is reduced. Thus during the case creation stage, each question and answer description is associated with an ontological concept defined in the domain of discourse. This is similar to how [1] associates ontology concepts with pre-defined questions. In

PreDiCtS we want to make use of such <concept-question> association so that questions and answers are implicitly defined in a taxonomy. This association is also important when similarities between QAPairs and between cases are calculated.

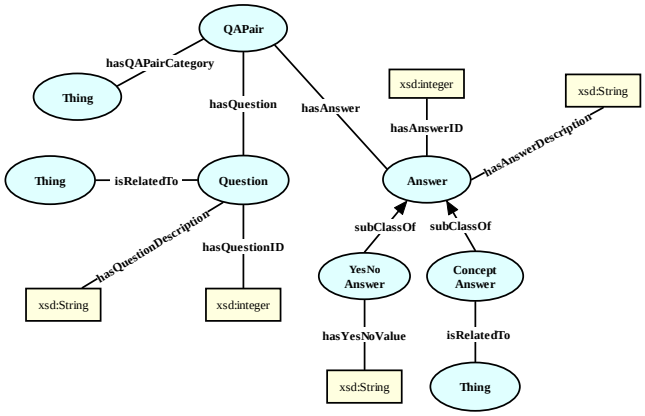

Fig.1: CCBROnto Problem structure

The Solution in PreDiCtS provides a hook where composition templates can be inserted. The main goal behind such a structure is to be able to present abstract composition knowledge as solutions to the user's request and at the same time allow for more flexibility when searching for actual services. In fact each *Solution* is defined to have an *Action* which has a description and is*DefinedBy* an *AbstractTemplate*. A template can be sub-classed by any service composition description, such as that defined by OWL-S. An OWL-S template in this case is an intersection between a service, profile and process definitions.

4. PreDiCtS: implementation issues

As explained in other sections, the PreDiCtS framework allows for the creation and retrieval of cases in its *Similarity phase* (see Fig. 2). The respective components that perform these two tasks are the *CaseCreator* and the *CaseRetrieval*. PreDiCtS is written in Java and is developed in Eclipse. It uses a MySQL database to store the cases and makes use of both Jena and the OWL-S APIs.

The *Similarity phase* is triggered by the user whenever she requires knowledge related to past compositions. In PreDiCtS the user is not expected to know exactly which type of services or service composition are required but she is required to answer a set of questions such that the system identifies more clearly what is required. Given information related to the domain, the retrieval process is initiated whereby all questions in a taxonomy relevant to that particular domain are presented to the user. Given the set of questions to choose from, the user can then decide to answer some of these questions. Depending on the answers provided, the system will try to find cases

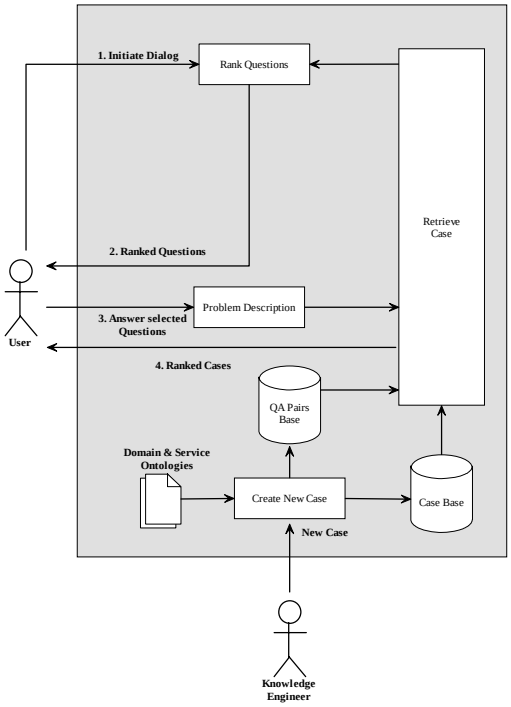

Fig.2: Taxonomic CCBR in PreDiCtS (adapted from Weber03)

in which questions where answered in a similar manner. A similarity measure is used to rank cases. The questions which are present in the retrieved cases but which are still unanswered, yet are related to the problem, are then presented in a ranked order to the user. The process continues until the user either chooses a case which includes a suitable solution or else, in absence of such a case, decides to adapt one of the most similar cases, thus further personalising the solution to her needs. The user can also opt to create a case from scratch to meet her requirements.

In the next sections we will describe the above mentioned PreDiCtS components by referring to an example from the health domain which deals with the combination of services that are used when a patient is admitted to hospital.

3.2 Case Creation

The CaseCreator component allows the expert user to add a new case to the case base. A case *c* can be defined as $c = (dsc, cxt, \{q_1a_3 \ldots q_ia_j\}, act, frq)$ where;
dsc is a textual description of the case.
cxt represents a set of context related features, such as *Role* and *CaseCreator* information based on foaf.
$\{q_1a_3 \ldots q_ia_j\}$ is a representation of the problem state by a set of question-answer pairs

act denotes the solution which is represented by service composition knowledge stored in an abstract template.
frq, is the frequency with which a case is reused.

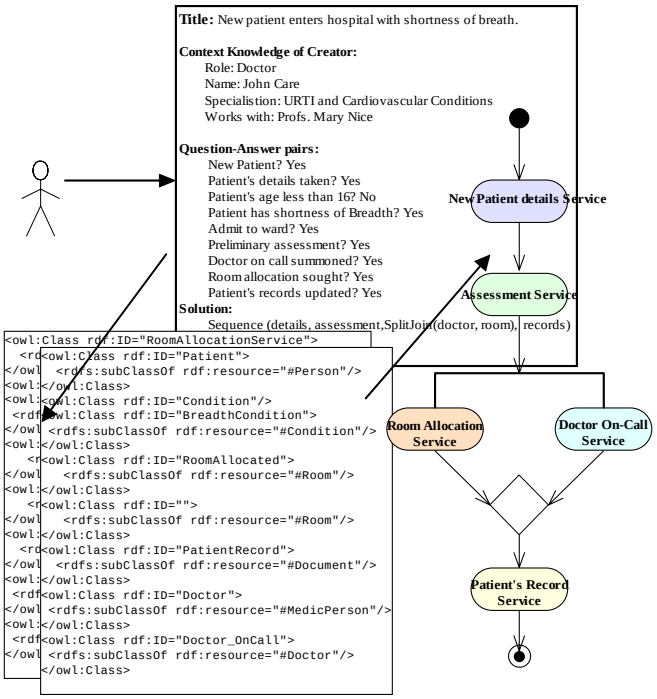

Fig. 3: Adding a new case

The example presented in Fig. 3 represents the combination of knowledge that is required to build a new case. PreDiCtS takes into consideration both domain and composition knowledge and combines them, based on the knowledge of the creator. In the example, the case creator is a Doctor (John) who specialises in URTI (Upper Respiratory Tract Infections) and cardiovascular conditions. The case in question represents the situation where a new patient, who is more then 16 years old, has entered hospital with shortness of breath. The creator enters context information about himself and any relations that he has with other persons. In this scenario, John has work relations with Professor Mary Nice. This information provides for a level of reputation in the expertise of the creator. The composition knowledge in this case represents a number of services that the hospital system wants to use to efficiently cater for patients entering hospital. This particular functionality is required to monitor the patient from the moment that he enters the hospital until he is comfortably stationed in a room.

To add service information to a case, the creator can use a visual component which is based on UML activity diagrams, though other representations, which are more user-friendly, are being considered. Each visual representation is mapped into a

process model representation. In this work we use OWL-S as the underlying language for this representation.

A *service* definition in OWL-S is just a place holder for information relating the profile, process and grounding. We are not considering any grounding knowledge at this stage, since this will be tackled later on in the *Integration phase* when actual service bindings are sought. As regards the profile, we only consider that knowledge which is relevant and which is not tide to specific providers. The profile part of the template includes the definitions of *inputs* and *outputs*, *profilehierarchy* and references to the process and service components. The profile hierarchy is considered to be of particular importance since it represents a reference to the service domain knowledge, that is, it identifies the taxonomic location of a particular set of service profiles. We think that such ontologies will become increasingly more important in relation to best practice knowledge. The template also provides information related to how a number of service components are combined together. What is most important here, are the control constructs such as *Sequence*, *If-Then-Else*, and *Split* that determine the order of execution of the service components. These service components are defined through the OWL-S *Perform* construct which associates a particular service component with another by binding its outputs to another service component's inputs.

An important aspect of case-creation in CCBR is the addition of question-answer pairs since they are fundamental for the case retrieval process. Through PreDiCtS we allow the creator to either reuse existing QA pairs or create new ones. Textual questions are associated with concepts defined in ontologies and this provides an implicit taxonomic structure for QA pairs. Such association provides the possibility to reason about these concepts, and also to limit the number of questions to present to the user during the retrieval process. The taxonomic theory requires that each case includes the most specific QA pair from a particular taxonomy. Given the open-world assumed by ontologies on the Web, we assume that the knowledge (triples) associated with a set of QA pairs is closed by adapting the idea of a local-closed world defined by [12].

Adding a new case to the case base is mainly the job of the knowledge expert, nevertheless we envision that even the not so expert user may be able to add cases when required. For this reason we have used the same technique as that used by recommending systems and also adopted by [21], which allows case-users to give feedback on the utility of a particular case to solve a specific problem.

3.3 Case Retrieval

Similarity is based on an adaptation of the taxonomic theory, and is divided into two steps, similarity between question-answer pairs and an aggregate similarity to retrieve the most suitable cases. The prior, involves the similarity between the QA pairs chosen by the user and those found in a case. In the taxonomic theory two pairs are defined to be more similar if the one found in the case is a descendant (therefore more specific) of the other, rather then its parent (therefore more generic). Though we have adopted this similarity assessment metric, we take into consideration that each QA

pair is a set of triples or rather an acyclic directed graph. Thus similarity between QA pairs is based on the similarity between two such graphs. The taxonomic similarity is calculated as follows:

$$sim\ (C_{Q1}, C_{Q2}) = \begin{cases} 1 & if\ C_{Q2} \subseteq C_{Q1} \\ (n+1-m)/(n+1+m) & if\ C_{Q1} \subseteq C_{Q2} \\ 0 & otherwise \end{cases}$$

where, C_{Q1} and C_{Q2} are concepts

n= number of edges between C_{Q1} and the root i.e. the concept *Thing*

m= number of edges between C_{Q1} and C_{Q2}

Having calculated such similarity between QA pairs then an aggregate similarity metric is used to calculate the overall similarity between the user query Q_U and a case problem description, P_C. This aggregate similarity is calculated as follows:

$$sim\ (Q_U, P_C) = \frac{\sum_{i \in Q_U, j \in P_C} sim(C_{Q\,i}, C_{Q\,j})}{T}$$

where, T in the original taxonomic theory represents the number of taxonomies, here it represents the number of different ontologies that are used to define the concepts found in the QA pairs.

We are also looking at other research work which provides for similar measures, in particular work related to ontology-based similarity measures [13], [16] and semantic distance [5], [14]. Such work is important since it does not only consider the taxonomic similarity between concepts but also similarity based on the number of relations and attributes associated with the concepts.

4. Conclusion

In this paper we presented the main concepts behind PreDiCtS. The use of CCBR as a pre-process to the service discovery and composition is promising since it provides for inherent personalisation of the service request and thus as a consequence also more personalised compositions. We also presented CCBROnto as a case definition language which allows for seamless integration between CCBR and the Semantic Web, by providing reasoning capabilities about concepts within the case definitions. Nevertheless, there is still a lot to be done, especially where it comes to case generation and evaluation. A case base can only be evaluated effectively if the number of cases is large. We are infact considering the possibility of generating cases, for experimental purposes, by extracting the required template knowledge from already available service descriptions and then adding context information and QA pairs. Other issues for future consideration include the design of the questions and the

way in which they are associated with ontology concepts, the effective evaluation of the similarity metrics used with an eye on work being done on semantic similarity and also the inclusion of an adaptation component. The latter will provide for more personalisation of the solutions presented by PreDiCtS and thus also of the services that will be presented to the user.

References

1. A.Aamodt, M. Gu, A Knowledge-Intensive Method for Conversational CBR, Proc. ICCBR'05, Chicago, August 2005
2. C. Abela, CCBROnto, http://www.semantech.org/ontologies/CCBROnto.owl
3. D.W Aha, L.A. Breslow, H. Muñoz-Avila, Conversational case-based reasoning. Applied Intelligence, 14, 9-32. (2001).
4. M.S. Aktas, D.B. Leake. et al, A Web based CCBR Recommender System for Ontology aided Metadata Discovery, GRID'04
5. A. Bernstein, et al, Simpack: a Generic Java library for Similarity Measures in Ontologies, University of Zurich, August 2005.
6. F. Bry et al, Context Modeling in OWL for Smart Buildings, Proc. of GvB2005.
7. M. d'Aquin, et al, Decentralized Case-Based Reasoning for the Semantic Web, Proc. ISWC 2005
8. A. Dey, Understanding and Using Context, in proceeding of Personal and Ubiquitous Computing, issue on Situated Interaction and Ubiquitous Computing, Feb 2001.
9. B. Diaz-Agudo et al, On Developing a Distributed CBR Framework through Semantic Web Services, Workshop on OWL: Experiences and Directions, Galway'05
10. A. Goderis et al. Seven bottlenecks to workflow reuse and repurposing. 4th Int. Semantic Web Conference, Galway, Ireland, 6-10 Nov. 2005
11. K. Gupta, Taxonomic Conversational Case-Based Reasoning, Proceedings of the 4th International Conference on Case-Based Reasoning, 2001
12. J. Heflin, H. Muñoz-Avila, LCW-Based Agent Planning for the Semantic Web, AAAI Workshop WS-02-112002
13. M. Hefke, A Framework for the successful Introduction of KM using CBR and the Semantic Web Technologies, I-Know 2004
14. N.Henze, M. Herrlich, The Personal Reader: A Framework for Enabling Personalization Services on the Semantic Web, Proc. of ABIS 04, Berlin, Germany.
15. Z. Maamar et al, Context for Personalised Web Services, 38th Hawaii International Conference on system Science, 2005
16. A. Maedche, V. Zacharias, Clustering Ontology-based Metadata in the Semantic Web, Joint Conferences (ECML'02) and (PKDD'02), Finland, Helsinki, 2002
17. J. Peer, A POP-based Replanning Agent for Automatic Web Service Composition ESWC'05
18. J. Scicluna et al, Visual Modelling of OWL-S Services, IADIS International Conference WWW/Internet, Madrid Spain, October 2004
19. E. Sirin et al, Semi-automatic composition of web services using semantic descriptions, in ICEIS 2003, Angers, France, April 2003
20. E. Sirin et al. HTN planning for web service composition using SHOP2. Journal of Web Semantics, 1(4):377-396, 2004
21. B.Weber, S. Rinderle, W. Wild, M. Reichert, CCBR–Driven Business Process Evolution, Proc. ICCBR'05, Chicago, August 2005