jUDDI+: A Semantic Web Services Registry enabling Semantic Discovery and Composition

Francesco Colasuonno, Stefano Coppi†, Azzurra Ragone†, Luca L. Scorcia,
Tommaso Di Noia†, Eugenio Di Sciascio†
SisInf Lab – Politecnico di Bari – Via Re David, 200, I-70125, Bari, Italy
fcolasuonno@gmail.com, luca@skakkinostri.it, {s.coppi,a.ragone,t.dinoia,disciascio}@poliba.it†

Abstract

We propose an extended version of a UDDI registry for semantic-based automated Web service discovery and composition, compliant with Semantic Web technologies. The approach exploits non-standard inference services in a fragment of OWL DL to perform semantic matchmaking and covering. The proposed approach also deals with effect duplication and non-exact solutions, computing an approximate composition and providing an explanation of which part of the request is not covered by the composite web service.

1 jUDDI+

jUDDI+ is an extended version of the open source UDDI implementation by the Apache Software Foundation – jUDDI[1] – able to cope with OWL based annotation of Web Services (WSs). The semantics of the WS description is exploited in order to perform both a semantic-based discovery of a single WS and a semantic-based WS composition with respect to a given request. Both the algorithms for discovery and composition [6] are based on non-standard Concept Abduction for Description Logics [1].

1.1 jUDDI+ Approach to Composition

Differently from other pure-semantic based WS composers, the approach implemented in jUDDI+ (whose basic version is proposed in [6]) does not try to discover "simple" subsumption relations between preconditions and effects (or inputs and outputs) of a sequence of WSs – contravariance [5]– but it is also able to cope with (a) non-exact matches and (b) effects duplication. (a) If it not possible to compute a sequence of WSs satisfying all the user requests, then an approximate solution is computed based on the semantics of both the WS descriptions (profile) and the request. An explanation of what remains uncovered by the computed composite WS is also provided, as a result of the composition process. For instance, if the user is looking for a double room in a hotel with SPA and swimming pool, if nothing better is available, maybe the user could be interested also in a hotel without a swimming pool. (b) In order to execute a WS, its preconditions/inputs must be satisfied, possibly using information provided by other services. Moreover care has to be paid in avoiding the duplication of effects/outputs when composing services, which might be also due to entailment relationships among different effects provided by services being composed (see Figure 1). After the execution of the composite service we do not want two or more services providing the same features, even partially overlapping. Turning to the classical scenario proposed in http://www.org/TR/ws-arch-scenarios/, a booking agent organizing a trip and composing two services, one able to book both *a hotel stay and a flight* and another *a flight and a car rental*, would not be much appreciated if its outcome is two flights booked for the same trip, together with the hotel and the car.

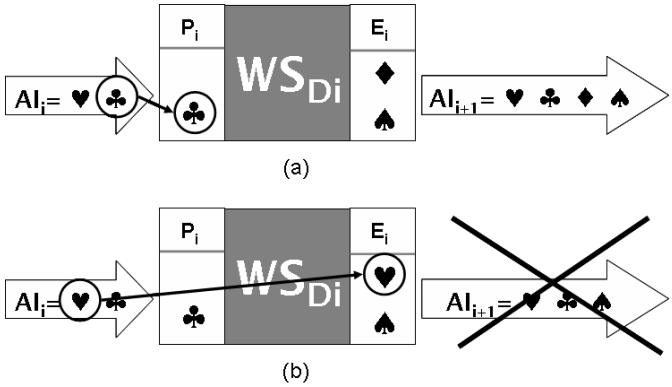

Figure 1. The role of preconditions and effects during web service composition.

[1] http://ws.apache.org/juddi/

1.2 Architecture for Discovery and Composition

A schematic of the jUDDI+ architecture is depicted in Figure 2. The main component of jUDDI+ is the *Web Service Orchestrator* – WSO. It uses MaMaS-tng[2], a Description Logics based reasoner, exposing non-standard inference services, able to cope with incomplete information in a matchmaking scenario [2]. In the proposed setting,

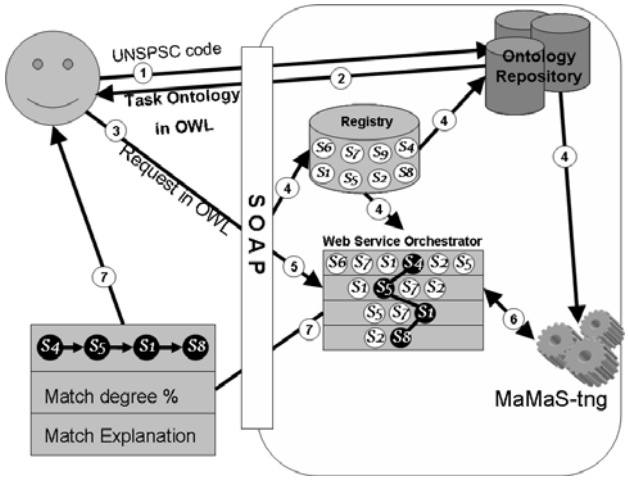

Figure 2. Schematic of jUDDI+

a user request is a triple $\langle P_0, E, \sigma \rangle$, where P_0 represents the initial preconditions/inputs provided by the user, *e.g.*, VALID_CREDIT_CARD, E the desired effects/outpus after the WS(s) execution, *e.g.*, flight reservation and double room booking in a hotel with a swimming pool and SPA, σ is a two-values variable in the set {discovery, composition}: if $\sigma =$ discovery then a discovery process is performed, a composition one in the $\sigma =$ composition case.

In the OWL-S (http://www.daml.org/services/owl-s/1.1/) compliant version of jUDDI+, the composition process is performed in the following steps:

1. The agent determines the task ontology that will be used (or the relative URL), selecting it via a UNSPSC code.

2. The selected ontology (or the relative URL) is sent back to the agent.

3. Using the selected ontology, the user formulates a request, representing her desired effects E, using its OWL Classes and Properties. Finally the user selects if she is interested in a composite WS or in discovering appealing single WSs. The request is then transmitted to the system, together with the initial preconditions P_0 the user is able to provide. From the information provided by the agent, the system selects two types of information: (a) the UNSPSC code identifying the ontology and (b) the OWL request description and the initial preconditions.

4. Using the identifier of the ontology – (a) in step 3) – the system retrieves the corresponding web services instances and sends them to WSO and the corresponding task ontology to MaMaS-tng.

5. The requested effects and the initial preconditions provided buy the user –selected in (b) part of step 3)– are sent to WSO.

6. The information is converted to DIG[4] syntax and sent to MaMaS-tng.

7. (a) If the user asks for a composite service able to satisfy her request, then WSO computes: the composite service, the matching degree and, when only an approximate solution is available, an explanation of which part of the request is not covered by the composite service (*i.e.*, an explanation of why the covering is not complete), using the algorithms proposed in [6]. All these information are returned to the requester. The matching degree is computed using semantic similarity metrics[3].

   (b) If the user asks for single services able to satisfy her request, then WSO computes – for each WS referring to the same ontology – a semantic-based match degree vs. the request. In particular, for each service, a subsumption relation between the user provided preconditions/inputs and WS preconditions/inputs is checked. If the relation holds, the WS is selected as a candidate to satisfy the user request. For each candidate a subsumption relation between the WS effects/outputs and the ones requested by the user is also checked. If this subsumption holds, then the corresponding WS is ranked as one of the best matching ones. In this case we have that all the preconditions/inputs of the WS are satisfied by the provided ones and all the requested outputs/effects are satisfied by the ones computed/provided by the WS. If the second subsumption relation (between effects/outputs) does not hold, then the effects/outputs semantic-based covering degree and explanation are computed – via MaMaS-tng– as a solution to a Concept Abduction Problem. Hence, the set of candidate WSs is ranked with respect to the covering degree and the ranked list endowed with a semantic-based explanation of the match is provided to the user.

[2] http://sisinflab.poliba.it/MAMAS-tng/

1.3 jUDDI+ implementation for the EEE'06 Challenge

In the jUDDI+ version customized for the EEE'06 challenge (see Figure 3) we have extended the architecture with a module able to map an XML-Schema into a DIG ontology and the XML-based WS inputs and outputs descriptions into the corresponding DIG statements. At bootstrap time, a batch procedure maps all the XML-Schemas in their related DIG ontologies/taxonomies and WS inputs and outputs descriptions, stored within the registry, in DIG statements. When a user formulates the XML-request for a (composite) WS, it is converted into the corresponding DIG format. Then the above general steps can be reformulated as:

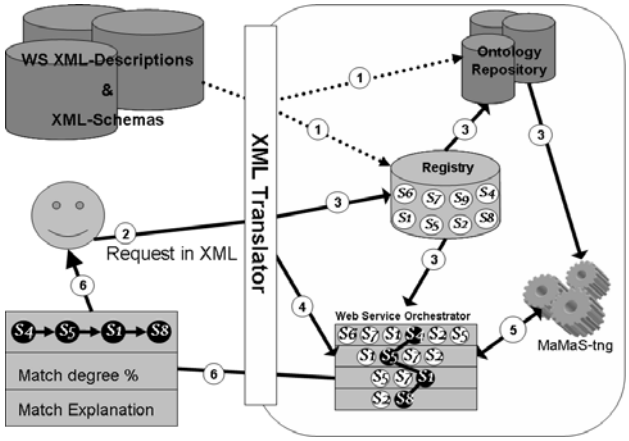

Figure 3. jUDDI+ **implementation for the EEE'06 Challenge**

1. At bootstrap time, all the XML-Schema and WS descriptions are imported from the repository and translated into the DIG language needed in order to communicate with MaMaS-tng. All the translated XML-Schema - ontologies - are then stored within the ontology repository and all the translated WS descriptions are stored within the registry. This step is executed only once (this is the reason for the dotted line in Figure 3) when the system is bootstrapped and is necessary in order to both translate all the needed information into the MaMaS-tng reasoner communication language and pre-classify the WS descriptions with respect to their input/output.

2. The user sends an XML-based request which refers to one of the XML-Schemas imported during the previous step.

3. All the WS descriptions in the repository referring to the request XML-Schema, are selected and sent to the WSO. Based on the XML-Schema URI, the related ontology is selected form within the ontology repository and sent to MaMaS-tng.

4. The request is translated into the corresponding DIG statements and sent to WSO.

5. The discovery or composition process is performed with the aid of the reasoner.

6. Results are sent back to the user. If the user asked for a discovery process, then the match degree is used to compute a ranked list of candidate WS descriptions matching the request. In case of composition, either the WSs covering the request are returned as the covered set to the user, together with the match degree and an explanation of unavailable parts in order to warn her that the composed service does not completely cover the request.

Notice that in this implementation of jUDDI+, the only translation to be performed at run-time is the one during step 4. All the translations needed during step 1 are executed only when the EEE'06 customized jUDDI+ is started for the first time.

References

[1] F. Baader, D. Calvanese, D. Mc Guinness, D. Nardi, and P. Patel-Schneider. *The Description Logic Handbook*. Cambridge University Press, 2002.

[2] S. Colucci, T. Di Noia, E. Di Sciascio, F. M. Donini, and M. Mongiello. Concept abduction and contraction for semantic-based discovery of matches and negotiation spaces in an e-marketplace. *Electronic Commerce Research and Applications*, 4(4):345–361, 2005.

[3] T. Di Noia, E. Di Sciascio, F. M. Donini, and M. Mongiello. A system for principled matchmaking in an electronic marketplace. *International Journal of Electronic Commerce*, 8(4):9–37, 2004.

[4] I. Dickinson. Implementation experience with the dig 1.1 specification. Technical Report HPL-2004-85, HP Laboratories, 2004.

[5] M. C. Jaeger, G. Rojec-Goldmann, C. Liebetruth, G. Muhl, and K. Geihs. Ranked matching for service descriptions using owl-s. *KiVS*, pages 91–102, 2005.

[6] A. Ragone, T. Di Noia, E. Di Sciascio, F. M. Donini, and S. Colucci. Fully automated web services orchestration in a resource retrieval scenario. In *Proc. of IEEE International Conference on web Services (ICWS 05)*, pages 427–434. IEEE, 2005.