Filters for Semantic Service Composition in Service-oriented Multiagent Systems

Alberto Fernández
Universidad Rey Juan Carlos
Tulipán s/n, 28933
Móstoles (Madrid) - Spain
+34 91 488 7084
alberto.fernandez@urjc.es

Sascha Ossowski
Universidad Rey Juan Carlos
Tulipán s/n, 28933
Móstoles (Madrid) - Spain
+34 91 664 7485
sascha.ossowski@urjc.es

ABSTRACT
In Service-Oriented MAS middle-agents provide different kinds of matchmaking functionalities. If no adequate services are available for a specific request, a planning functionality can be used to build up *composite* services. In order to take advantage of recent advances in the field of AI planning for this purpose, we propose exploiting organisational information of Service-Oriented MAS to heuristically filter out those services that are probably irrelevant to the planning process. We present a novel framework for service-class based filtering and show how it can be instantiated to a particular MAS domain based on organisational information.

Categories and Subject Descriptors
I.2.m [**Artificial Intelligence**]: Miscellaneous

General Terms
Design.

Keywords
Semantic Web Services, Multiagent systems, Service composition.

1. INTRODUCTION
Services are computational entities that can be described, published, discovered, orchestrated and invoked by other software entities. When the management of services is realised by agents, the term service-oriented MAS has become popular [5]. Service-oriented architectures usually include directory services that service providers register with. A request for a service with desired characteristics often results in a matchmaking process based on the profiles stored in the directory. In service-oriented MAS, this functionality is typically offered by middle-agents.

Even if no adequate services are found in this manner, it is still possible to build up *composite* services by purposefully combining several pre-existing services. The service composition planning problem has subtle differences with the classical AI planning problems, in particular as service composition plans need not be very deep but, in turn, can be built up from a vast number of services (operators) that are usually registered in the directory. This is particularly the case in open large-scale service-oriented MAS, where a pure AI planning approach can become impracticable.

In order to still take advantage of the recent advances in the field of AI planning, we suggest to reduce the set of input services that are passed on to an agent's composition planning component. In Section 2 of this paper we present a framework to heuristically filter out those services that are probably irrelevant to the planning process. Section 3 sketches how such filters can exploit information about the organisational structure underlying a MAS. Section 4 briefly discusses how our service composition filters are used in the European IST project CASCOM [1].

AAMAS'07, May 14-18, 2007, Honolulu, Hawai'i, USA.

2. GENERIC FILTERING FRAMEWORK
At a high level of abstraction, the service composition planning problem can be conceived as follows: let $P = \{p_1, p_2, \ldots, p_m\}$ be the set of all possible plans (composite services) for a given service request R, and $D = \{s_1, s_2, \ldots, s_n\}$ the set of input services for the proper service composition planner (i.e. the directory available). The objective of a filter F is to select a given number l of services from D, such that the search space is reduced, but the best plan of P can still be found.

The filter should make sure that the pruning of the search space for the planner is minimal. Put in another way: the bigger the subset of plans $P' \subset P$ that the planner can choose from, the bigger the probability that the plan of maximum quality is among them. A good heuristic to this respect is based on *plan dimension* and on the *number of occurrence*s of services in plans: a service is supposed to be the more important, the bigger the number of plans from P that it is necessary for, and the shorter the plans from P that it is required for. We can approximate this information by storing and processing the plans historically created. So, in principle, we can build up matrices that store, for every possible query, the number of plans in which each service appeared, classified by each plan dimension.

However, it soon becomes apparent that the number of services and possible queries is too big to build up all matrices of the above type. Furthermore, the continuous repetition of a very same service request R is rather unlikely. Finally, such an approach would not be appropriate when a new service request (not planned before) is required (which, in fact, is quite usual). To overcome this drawback, we assume the availability of *service class*

978-81-904262-7-5 (RPS)

information, so as to cluster services based on certain properties. If the number of classes is not too big, the aforementioned approach becomes feasible computationally.

Figure 1 depicts the structure of our approach to service composition filtering. With each outcome of a service composition request, a Historical Information Matrix H is updated. Setting out from this information, a Relevance Matrix v is revised and refined. Based on this matrix, service relevance can be determined in a straightforward manner. For each service composition request, the filtering method is based on this estimated service relevance function.

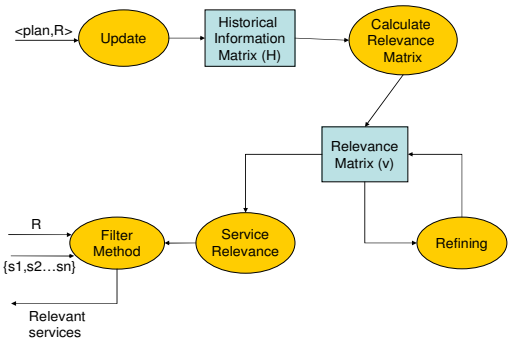

Figure 1 – Architecture of the filter component

2.1 Computing Filter Information

The *Historical Information Matrix* for a service class r compiles relevant characteristics of plans (composite services) that were created in the past in response to requests for services belonging to that class. In particular, for each plan dimension i and service class C it records the number of plans of length i that made use of services of class C. Historical Information Matrixes are updated as newly generated plans come in. If the service request is a logical formulae (given in disjunctive normal form), we *distribute* the contribution of the resulting plan among the affected Historical Information Matrixes.

As commented above, by ranking services we try to select a set of services that cover the largest subset of the plan space, as an attempt to maximise the chance of the best plan to be contained in it. Services that formed smaller plans in the past are considered more relevant, since it is easier to cover small plans that large ones, so with less services more plans can be covered.

The Relevance Matrix specifies the relevance of a service class s to be part of a plan (composite service) that matches the query for a certain service class r. We use the following function to aggregate the information about plans contained in the *Historical Information Matrixes* (remember that all this information is about a single request class r):

$$\text{Relevance}(s,r) = \frac{\sum_{d=1}^{m} \frac{n_d}{d^c}}{\sum_{d=1}^{m} \frac{N_d}{d^c}},$$

Where d is the dimension of the plan, m is the dimension of the longest plan stored, n_d is the number of times that s was part of a composite plan of dimension d for the request r, and N_d is the total number of plans of dimension d for that request. c is a constant > 0 that allows giving more importance to plans of smaller dimension (a straightforward value is c=1). Only dimensions with more than 0 plans are considered. With this calculus we obtain a *relevance* value between 0 and 1 for every given service class s with respect to the composition of a service of class r.

The Relevance Matrix $v(s,r)$ can be further refined in order to take transitivity into account. Consider the following situation: A plan that achieves C_1 is searched for, and that a potential solution is to compose the services C_2 and C_3 ($C_2 \oplus C_3$ for short). However, there is no service provider for C_3, but instead C_3 can be composed as $C_4 \oplus C_5 \oplus C_6$, so the final plan is $C_2 \oplus C_4 \oplus C_5 \oplus C_6$. Unfortunately, the value $v(C_4,C_1)$ is low and the service providing C_4 is discarded and not taken into account in the planning process, so the aforementioned plan cannot be found by the planner. Therefore, we will refine the relevance matrix by taking *transitivity* into account, e.g. through the following update: $v(C_4,C_1) = v(C_4,C_3)*v(C_3,C_1)$. The same holds for third-level dependencies (e.g.: $v(C_7,C_1) = v(C_7,C_4)*v(C_4,C_3)*v(C_3,C_1)$). This example motivates the definition of the $v^k(s,r)$ as a k-step relevance matrix

$$v^1(s,r) = v(s,r)$$

$$v^k(s,r) = Max\,(v^{k-1}(s,r),\ v^{k-1}(s,s_1) * v^{k-1}(s_1,r),\ v^{k-1}(s,s_2) * v^{k-1}(s_2,r),\ \ldots,\ v^{k-1}(s,s_n) * v^{k-1}(s_m,r))$$

As shown in the equation, we use the product as combination function and the maximum to aggregate the results. Note that the higher the value of k the better the estimation of the relevance of service classes. The refinement of the relevance matrix is repeated until it converges or until a timeout occurs.

The elevated time complexity of $O(n^3)$ for each refinement step is attenuated by the *anytime properties* of the approximation algorithm. Furthermore, recall that the number of classes n is supposed to be fixed and not overly high. Finally, note that several updates and refinements of the Relevance Matrix can be combined into a "batch" to be executed altogether when the system's workload is low.

There are several ways of obtaining the initial relevance matrix. If there are historical records of plans they can be used to calculate the matrix. Also, an a priori distribution can be assigned using expert (heuristic) knowledge. Still, the simplest solution is to let the service composition planning component work without filtering services until the number of plans generated is representative enough to start computing and refining the matrixes.

2.2 Using a Filter

The first step to calculate the *relevance* of a service s for a request r is the mapping of both to *classes of services*. Then, the relevance between the classes is calculated. We will use $v(s,r)$ to represent the relevance of *class s* for the *class r* in the request, and $V(S,R)$ as the *relevance* of service S for the service request R.

Considering that, in general, the service S belongs to *several* classes (s_1, s_2 … s_n), if a request R only includes a class (r) in its description: $V(S,R) = \max(v(s_1,r), v(s_2,r), \ldots, v(s_n,r))$.

However, if the request specifies a logical expression containing several classes of services (r_1, r_2 … r_m), we evaluate logical formulas using the *maximum* for disjunctions and the *minimum* for conjunctions; and inside the *maximum* is used to aggregate the service classes specified by the provider. For example, if the request R includes the formula $r_1 \vee (r_2 \wedge r_3)$, and the service S belongs to the classes s_1 *and* s_2, the calculus is as follows:

$$V(S,R) = \max[\ \max(v(s_1,r_1),\ v(s_2,r_1)),\ \min(\max(v(s_1,r_2),\ v(s_2,r_2)),\ \max(v(s_1,r_3),v(s_2,r_3)))\ ].$$

We have build three different types of filters based on the above approach, depending on whether they return only services whose relevance exceeds a certain *threshold*, whether they are among the *k best* services, or whether they are within a certain *percentage* of the most relevant services. In addition, every such filter allows specifying a certain probability by which services are selected randomly to assure an adequate exploration of the plan space.

3. ROLE-BASED FILTERING

In many service-oriented systems, agents are conceived as mere wrappers for web services. However, agents are not only able to execute a service but may also engage in different *types of interaction* related to that service, in the course of which they play several *roles*. For example, in a medical emergency assistance scenario, an agent providing a *second opinion* service should not only be able to provide a diagnostic; it may also be required to explain it, give more details, recommend a treatment, etc. Therefore, a service provider may need to engage in several different interactions, and play a variety of different *roles*, during the provision of a service.

Our role-based filtering method relies on taxonomies of roles and type of interactions to determine service classes. The idea is to relate roles searched in the query to roles played by agents in the composite service, that is, what are the roles typically involved in a plan when a role r is included in the query. For example, it is common that a *medical assistance* service includes *travel arrangement, arrival notification, hospital log-in, medical information exchange* and *second opinion* interactions.

In the medical emergency assistance domain addressed by the European IST project CASCOM [1], for instance, each service provider advertises a set of possible roles from the role ontology that it can play. Similarly, in service requests it is allowed to specify the roles searched from the role ontology as a logical expression in disjunctive normal form. By establishing a mapping from the elements of the role ontology to service classes, the above filtering framework becomes applicable. A detailed description of our approach to role-based service coordination can be found in [2].

4. CONCLUSIONS

There is currently a limited number of composition planning approaches based on OWL-S or DAML-S (e.g. [6], [7], [8]). However, they are geared towards standard service-oriented architectures. To the best of our knowledge none of them makes use of organisational information [9] within a service-oriented MAS to filter services.

We have implemented our role-based filter mechanism in JAVA on top of a P2P extension of the JADE agent platform [4]. Together with OWLS-XPlan [6], a heuristic hybrid search AI planner for the composition of OWL-S services, it is part of the Service Composition Planning Agent (SCPA), a key part of the CASCOM abstract architecture [3].

The performance of the CASCOM approach in general, and the adequacy of the SCPA composition planning approach based on an AI planner and configurable, adaptive, service class based composition filters in particular, will be evaluated during a field trial in the medical emergency assistance domain. We will also compare the performance of our role-based method to other semantic service composition filter instantiations, in particular, those based on OWL-S Service Category information.

5. ACKNOWLEDGMENTS

This work has been partially supported in part by the European Commission under grant FP6-IST-511632 (CASCOM), and by the Spanish Ministry of Education and Science through projects TIC2003-08763-C02-02 and TIN2006-14360-C03-02.

6. REFERENCES

[1] Cáceres, C., Fernández, A. and Ossowski, S. CASCOM – Context-aware Health-Care Service Coordination in Mobile Computing Environments. *ERCIM News* 60 (2005), pp 77-78

[2] Cáceres, C., Fernández, A., Ossowski and S., Vasirani, M. Agent-Based Semantic Service Discovery for Healthcare: An Organizational Approach, *IEEE Intelligent Systems*, vol. 21, no. 6, pp. 11-20, Nov/Dec, 2006.

[3] Cáceres, C., Fernández, A., Ossowski, S. and Vasirani, M.: *An abstract architecture for semantic service coordination in agent-based intelligent peer-to-peer environments.* The 20th ACM 2006 Annual Symposium on Applied Computing (SAC-2006). Dijon (France).

[4] CASCOM Consortium. CASCOM Project Deliverable D4.1: IP2P Network Architecture. 2006

[5] Huhns, M. N., et al: Research Directions for Service-Oriented Multiagent Systems. *IEEE Internet Computing*, 9 (6), 2005.

[6] Klusch, M., Gerber, A., and Schmidt, M. Semantic Web Service Composition Planning with OWLS-XPlan. *Proceedings 1st Intl. AAAI Fall Symposium on Agents and the Semantic Web* (Arlington VA, USA, 2005).

[7] M. Sheshagiri, M. desJardins, and T. Finin. A planner for composing services described in DAML-S. *Proceedings of AAMAS 2003 Workshop on Web Services and Agent-Based Engineering*, 2003

[8] D. Wu, B. Parsia, E. Sirin, J. Hendler, and D. Nau. Automating DAML-S web services composition using SHOP2. *Proceedings of the 2nd International Semantic Web Conference (ISWC2003),* pages 20–23 (Sanibel Island, Florida, USA, October 2003)

[9] Zambonelli, F., Jennings, N. R., and Wooldridge, M. Organizational Abstractions for the Analysis and Design of Multi-agent Systems. *Agent-Oriented Software Engineering: First International Workshop, AOSE 2000*, (Limerick, Ireland, June 10, 2000). 235-251.