Organizational Models for Semantic Service Coordination *

Alberto Fernández, Sascha Ossowski, Matteo Vasirani

University Rey Juan Carlos, Madrid, Spain
{alberto.fernandez,sascha.ossowski,matteo.vasirani}@urjc.es

Abstract. Service-Oriented Computing and agent technology are nowadays two of the most active research areas in distributed and open systems. Given the growing interest in bridging the two worlds, in this paper we describe a novel approach for service description, discovery and composition in multiagent systems, based on organizational models. An extension to service matching techniques will be presented, as well as a role-based filter mechanism for service composition.

1 Introduction

Service-Oriented Computing [5] is a novel computing paradigm that conceives services as basic elements to develop applications and systems. Services are self-describing, platform-independent computational entities that can be described, published, discovered, orchestrated and invoked by other software entities in order to support rapid and low-cost composition of distributed applications.

Agent technology provides designers with an interaction-centric way of designing open and distributed software systems [8]. There is a growing awareness that organizational models are fundamental to regulate open multi-agent systems, to instill desired properties and to provide coordination among the agents [10, 12].

Although these two research areas have different backgrounds and motivations, there is a growing interest in bridging the two worlds; from one side, software agents can be viewed as potential users and providers of Semantic Web Services, from the other side web service technology can be used to support the interactions in multiagent systems [3].

The paper is organized as follows: in section 2 we will present an agent-based architecture for service coordination in IP2P environments, which has been developed within the CASCOM 1 project, in section 3 and 4 we will discuss the benefits of organizational abstractions to cope with dynamic service discovery (section 5) and composition (section 6).

* This work has been partially supported in part by EU, MEC, CAM and URJC under grants FP6-IST-511632, TIC2003-08763-C02-02, TIN2006-14630-C03-02, and URJC-CM-2006-CET-0300
1 www.ist-cascom.org
2 The CASCOM architecture

The CASCOM architecture is based on a layered approach that deploys modern software technologies, like semantic web services and intelligent agents, in a mobile, peer-to-peer environment (see figure 1).

![CASCOM Architecture Diagram](image)

The **Networking Layer** is in charge of managing the heterogeneity of networks, providing a generic P2P network infrastructure. It offers an efficient and secure, FIPA-HTTP-based agent message communication channel, and agent execution environment for resource-constrained mobile devices. In addition, it provides distributed and federated directory services (DS), a low-level lookup of services advertised by service providers.

The **Service Coordination Layer** intends to be the agent-based bridge between application agents and semantic web services. The main functionalities are semantic service discovery, matchmaking, composition and execution.

Service Discovery Agents (SDA) manage the discovery of required services, handling both abstract service descriptions and concrete service groundings. SDAs act as agent interfaces with the web-accessible directory service (DS).

The service matchmaking is conceived as a fine-grained, syntactic and semantic matching between a requested service and the services stored in the DS. The service matchmaking functionality is provided by Service Matchmaking Agents (SMA), which take into consideration input and output concepts, precondition and effect logic expressions, and also non-functional properties of the service.
Service Composition and Planning Agents (SCPA) are capable of creating value-added composite services that match concrete service specifications, when there isn’t a single service that fulfills a user query, using a heuristic hybrid search planner, called OWLS-Xplan [7].

The execution of services, composite as well as atomic, is provided by Service Execution Agents (SEA). SEAs coordinate the execution of each atomic service that composes the plan, discovering appropriate available service providers for each atomic service.

The Context Subsystem works as a gateway of context information between the Networking and the Service Coordination Layers. Its main functionality is to discover, acquire and store useful context information (e.g. the geographical position or the user’s preferences).

The Security and Privacy Subsystem, also orthogonal to the Networking and the Service Coordination Layers, is responsible of ensuring security and privacy of information throughout the different components of the CASCOM infrastructure.

3 Organizational Abstractions

In recent years, social and organizational aspects of agency have become a major issue in research; organizational structures, norms, conventions, are now part of the design of open multi-agent systems.

Agents are not only able to execute the service, but also they can engage in different communicative interactions around that service. For example, consider a healthcare assistance scenario, where an agent providing a second opinion service should not only be able to provide a diagnostic, but it may also be required to explain it, give more details, recommend a treatment, etc. This means that the agent provides a set of interaction capabilities based on the service.

This interaction-centric view can be exploited in the service matchmaking. The efficiency of the matchmaking process can be improved for example by previously filtering out those services that are incompatible in the terms of interaction types. Also the precision of the matchmaking process can be enhanced by including interaction-centric information. For instance, a diagnosis service may require symptoms and medical records as input and produce a report as output. However, the service functionality can be achieved either (i) by actually generating the report, (ii) by retrieving a previously done or (iii) by a brokering service to contact other (external) healthcare experts. In all three cases the input and output might be the same, but the role the service plays in the corresponding interactions is different.

The service composition process can also be guided by taking into account the interaction context in which service provider agents participate, that is, what are the roles typically involved in a plan when a role \( r \) is included in the query.

In order to enrich web service descriptions with interaction-centric information, we use a subset of the RICA organizational model described in [10]. We first analyze different use cases of the application domain scenario, identifying the
types of social interactions as well as the roles that take part in those interactions. The next step is an abstraction process in which the social (domain-dependent) interactions are generalised into communicative (domain-independent) ones.

As an example, let’s consider a second opinion use case scenario in the healthcare domain. In this scenario, a physician of an emergency center asks an external health professional for a diagnosis on the basis of the symptoms and the medical records of the patient, like exams and past diseases.

The interaction between the patient and the health professional can be modeled by a sequence of (communicative) actions between the two agents involved. The physician asks the health professional for an opinion, providing the symptoms and the medical records. If there is insufficient information, the health professional requests additional information (possibly several times) and finally gives his advisement. If the provided diagnosis is not sufficiently clear, the patient can also solicit an explanation.

Such an analysis isolates 3 different interactions: (i) the second opinion exchange, which can comprise (ii) a detailed information exchange. When the second opinion exchange finishes, an explanation (iii) can occur.

In the same way, analysing different use case scenarios we have derived an ontology that contains a taxonomy of types of interactions, and a taxonomy of roles that take part in those interactions (see figure 2).

The top concept of the ontology is the Communication, which represents the most generic type of interaction. Any type of interaction is a specialization of this concept. The first level of specialisation is the ClosedActionPerforming interaction type, which represents any interaction that implies an action performing (like Advisement, Admission, InformationExchange...). New (generic and domain) interaction types can be incrementally be added to this taxonomy making it more complete and reusable.

![Fig. 2. Partial view of the interaction taxonomy](image-url)
4 Interaction-centric Service Descriptions

In this section we outline our proposal to integrate role-based information into service advertisements and service requests.

In a service advertisement we specify two kinds of information related to the interactions in which the service provider agent can engage:

1. **Provider Role**: is the role played in the interaction, e.g. the advisor role in the second opinion scenario;
2. **Depending Roles**: a set of roles that must be played by the requester agent for the correct accomplishment of the service. For instance, in an advisement interaction of a second opinion service, the provider may need to initiate an information exchange interaction in which it plays the **informee** role, and the requester plays the **informer** role. These roles are given by a formula in disjunctive normal form, i.e. a disjunction of conjunctions of roles.

A service request is a description of an abstract service that the user is searching. In the case of a service request, from an organizational point of view, we consider that a query comprises two elements:

1. **Searched Provider Roles**: although one role will be enough in most cases, we allow for more complex search patterns in which the provider is able to play more than one role. As in the case of service advertisements, we require an expression in disjunctive normal form.
2. **Requester Capability Roles**: a set of roles that define the capabilities of the requester (roles it is able to play). This information is important if the provider requires interaction capabilities from the requesters. For example, the requester of a second opinion can inform that it is able to provide information (informer) if needed.

Table 1 shows an abstract example of a service advertisement and query.

**Table 1.** Abstract example of a role-based service advertisement and query

<table>
<thead>
<tr>
<th>Advertisement</th>
<th>Depending Roles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provider Role</td>
<td></td>
</tr>
<tr>
<td>$R_1$</td>
<td>$(R_4 \land R_5) \lor R_6$</td>
</tr>
<tr>
<td>$R_2$</td>
<td>$(R_6 \land R_7 \land R_8) \lor (R_4 \land R_7)$</td>
</tr>
<tr>
<td>$R_3$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Query</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Searched Provider Roles</td>
<td>$R_2 \lor (R_6 \land R_4)$</td>
</tr>
<tr>
<td>Requester Capability Roles</td>
<td>$R_4, R_7, R_8$</td>
</tr>
</tbody>
</table>

We use the service description language OWL-S in our system. As organizational information does not fit in any of the predefined fields of the OWL-S
service profile (used for describing what the service does), we opted for including the role description as an additional parameter, called ServiceRoles in the case of service descriptions and QueryRoles for service requests.

5 Service Matchmaking

We have developed a role-based matching algorithm [4] that takes as inputs a service request (R) and a service advertisement (S), and returns the degree of match (dom) between them. The degree of match is a real number in the range [0,1], so that service providers can be selected by simply comparing these numbers.

The semantic match of two roles RA (advertisement) and RQ (query) is made based on the ontology of roles. It is a function that depends on two factors:

1. Level of match. This is the (subsumption) relation between the two concepts (RA, RQ) in the ontology. We differentiate among the four degrees of match proposed by Paolucci et al. [9]: exact (if RA = RQ), plug-in (if RA subsumes RQ), subsumes (if RQ subsumes RA), fail (otherwise).

2. The distance (number of arcs) between RA and RQ in the taxonomy.

We combine both criteria into a final degree of match, which is a real number in the range [0, 1]. In this combination, the level of match always has higher priority: the value representing the degree of match is equal to 1 in case of an exact match, it varies between 1 and 0.5 in case of a plug-in match, rests between 0.5 and 0 in case of a subsumes match, and it is equal to 0 in case of a fail.

We use the equation (1), where \( \| R_A, R_Q \| \) is the distance between RA and RQ (\( depth(R_A) - depth(R_Q) \)) in the role ontology (if there is a subsumption relation between them).

\[
dom(R_A, R_Q) = \begin{cases} 
1 & \text{if } R_A = R_Q \\
\frac{1}{2} + \frac{1}{2 \cdot \| R_A, R_Q \|} & \text{if } R_A \text{ is subclass of } R_Q \\
\frac{1}{2} \cdot e^{\| R_A, R_Q \|} & \text{if } R_Q \text{ is subclass of } R_A \\
0 & \text{otherwise}
\end{cases}
\] (1)

The semantic match between two services is done by searching the role in the advertisement S that best matches the one in the query (R). The degree of match between a role in the request and a service advertisement, given the set of capabilities of the requester, is done by comparing the searched role with every other given role and returns the maximum degree of match. For each role in the advertisement, the match between the provider roles is made, as well as the match between the depending roles and the capabilities of the requester. The minimum of both values is considered the degree of match. In case of logical expressions, the minimum is used as combination function for the values in a conjunction and the maximum for disjunctions (which always keep the value resulting of the combination within the range [0,1]).
6 Service Composition

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 organizational 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 role and interaction ontologies.

6.1 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 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 occurrences 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 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 3 depicts the structure of our approach to service composition filtering.

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 stores the number of plans of length \( i \) that made use of services of class \( C \). Historical Information Matrices are updated as newly generated plans come in. If the service request is a logical formula (given
in disjunctive normal form), we *distribute* the contribution of the resulting plan among the affected Historical Information Matrices.

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 aggregate the information about plans contained in the *Historical Information Matrices* so as to obtain a relevance value between 0 and 1 for every given service class $s$ with respect to the composition of a service of class $r$. In this calculus, the number of occurrences in plans and their dimensions are considered.

It might occur that a given service class $s$ is apparently not relevant for a service class $r$, based on historical information. However, $s$ is relevant for $s_i$ which, in turn, is relevant for $r$. Service class $s$ would probably be discarded and not taken into account in the planning process for $r$, so a plan cannot be found for $s$ if there is no service provider for $s_i$. The Relevance Matrix $v(s, r)$ is then *refined* in order to take this transitivity into account. We use the following $k$-step relevance matrix refining, which is repeated until it converges or a timeout is received.

$$
\begin{align*}
    v^1(s, r) &= v(s, r) \\
    v^k(s, r) &= \max\left(v^{k-1}(s, r), v^{k-1}(s, s_1) \cdot v^{k-1}(s_1, r), v^{k-1}(s, s_2) \cdot v^{k-1}(s_2, r), ..., v^{k-1}(s, s_n) \cdot v^{k-1}(s_n, r)\right)
\end{align*}
$$

(2)

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, the *relevance* of service $S$ for the service request $R$ is $V(S, R) = \max(v(s_1, r), v(s_2, r), ..., v(s_n, r))$. 

![Fig. 3. Architecture of the filter component](image)
However, if the request specifies a logical expression containing several classes of services \( (r_1, r_2, \ldots, r_m) \), 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.

Finally, a filter method defines which services are chosen as relevant 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.

### 6.2 Role-Based Filtering

The class-based semantic service composition filter described in the previous section can be refined based on the information provided by the role and interaction ontologies. The idea is to relate roles searched in the query to roles played by agents in the composite service, that is, 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 our approach (described in section 4) each service provider can advertise 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. Still, for the purpose of this paper, we just need to map each role from the ontology to a service class of the filtering framework.

### 7 Conclusions

In this paper, we have described a novel approach for service description, discovery and composition in multiagent systems, based on organizational models. We have identified roles and interaction ontologies as key information to this respect, and exploited in both service matchmaking and service composition.

An extension to general-purpose service matching techniques that exploits this additional organizational information has been put forward, which has given rise to the implementation of a role-based matchmaking component, which was combined with the OWLS-MX matchmaker [6]. Our experiments demonstrated that the combination of both matchmakers outperforms the OWLS-MX stand alone matchmaker in both efficiency and effectiveness (we used several measures proposed within TREC\(^2\), such as precision/recall curves, average precision and \( R \)-precision, details can be found in [4]).

We have presented a heuristic approach, based on plan dimension and past plans, to filter out those services which are probably irrelevant to the planning process. Since several services are filtered out before the planning is carried out, the completeness of the planning might be affected. However, this risk can be

\(^2\) http://trec.nist.gov/
admissible if the complexity of the planning is too high, as is expected to occur in large open multiagent systems. Filtering techniques for pruning the search space have been used in many planning algorithms, but they are usually integrated within the algorithm. In the case of semantic web service composition, few approaches have been presented. In [11], a human assistance tool (not automatic) is presented. [2] describes a directory which supports user-defined selection and ranking functions. Our approach is independent of the domain and planner engine used.

Currently we are evaluating this component as part of the trial phase of the CASCOM project. Future works will investigate how the matchmaker behaves if other aggregation functions are used instead of minimum and maximum to combine the degrees of match of individual roles, and also how quality of service and trust mechanisms [1] can be incorporated into our approach.

References