Role-based Infrastructures for Agents

Giacomo Cabri
Dipartimento di Scienze dell'Ingegneria – Università di Modena e Reggio Emilia
Via Vignolese, 905 – 41100 Modena – ITALY
E-mail: giacomo.cabri@unimo.it

Abstract

Multi-agent systems represent the most recent technology to design and develop distributed applications. Agents can also gain advantage from mobility, i.e., the capability of changing execution environment in world modeled as a set of nodes with different resources and services. This technology calls for appropriate local infrastructures to manage the interactions among agents and between agents and environments. We propose to build such infrastructures on the basis of roles, which represent the behavior and the capabilities of agents. Some examples are reported to explain the exposed concepts in concrete fields.

1 Introduction

Software agents are proposing as the future of distributed systems, and they will soon populate the most systems in the world, and in particular the Internet. They are able to perform tasks on behalf of users, due to their main features – *autonomy*, *proactiveness*, *reactivity* and *sociality* [6]. The complexity of the applications has lead to the division of the global task into smaller and simpler tasks, each one delegated to one agent. So applications are composed of several agents, and thus called multi-agent. In this scenario, the social behavior of the agents implies interactions among the agents cooperating in one application. However, in wide distributed systems, and in particular in the Internet, interactions can occur also among agents of different applications, which may have a competitive behavior, to gain the use of resources [5].

The feature of *mobility* [7], enhancing the autonomy of agents, implies further advantages. Generally, mobile agents can save bandwidth by moving locally to the sites where the resources are located, and do not rely on continuos network connections. Users are not required to be connected to the network continuously: they can send their agents, disconnect, and then reconnect when the agents have carried out their tasks to retrieve them.

Agent mobility is an important issue to be taken into consideration when designing distributed applications and, in particular, when defining the interactions with other entities. This paper takes into consideration mobile agents, but the presented ideas are also suitable to fixed agents.

This paper proposes to build infrastructures for agents based on the concept of role. A *role* is defined as the *behavior* and the *set of the capabilities* expected for the agent that plays such role. The concept of role has been exploited in the Object-Oriented field to design complex applications [9]. It recently started to appear in the agent area [8], even if several proposals are ad hoc solutions for given situations [2].

The paper is organized as follows. Section 2 introduces the concept of role for the agents. Section 3 presents the definition of infrastructures based on roles. Section 4 shows some examples. Section 5 concludes the paper and sketches some future work.

2 Agent Roles

In the agent scenario, a *role* can be defined as the *behavior* and the *set of the capabilities* expected for the agent that plays such role. This leads to a twofold viewpoint of the role: from the environment point of view, the role imposes a defined behavior to the entities that assumes it; from the application point of view, the role allows a set of capabilities, which can be exploited by agents to carry out their tasks. There are some characteristics of roles that distinguish them from the concept of agent. The role is *temporary*, since an agent may play it in a well-defined period of time or in a well-defined context. Roles are *generic*, in the sense that they are not tightly bound to a specific application, but they express general properties that can be used in different applications. Finally, roles are *related to contexts*, which means that each environment can impose its own rules

and can grant some local capabilities. As mentioned before, roles represent behaviors that agents are expected to show; who expects such behavior are entities external to agents themselves, mainly organizations [11] and environments.

3 Building Infrastructures

The key idea is that a set of roles determines a local infrastructure to be considered as intermediate between applications and environments. In the agent-based Internet applications there are not sharp delimitations between the actual application parts and the environments. Nevertheless, it is useful to separate some issues to simplify the design and to help the implementation and the deployment. We propose to model agent-based applications distinguishing four levels (see Figure 1):

- the *agent level* is the one where application agents live;
- the *infrastructure level* contains the roles that agents can assume in the environment; each site has its own set of admitted roles;
- the *policy & mechanism level* aims at defining the policies local to the environment and the mechanisms that implements the interaction among roles;
- finally, the *resource level* contains the local resources, such as information and services.

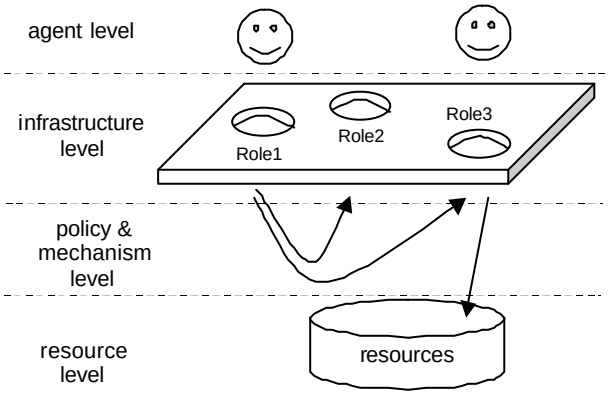

Figure 1. A role-based infrastructure

The second level can be considered as the interface of a site for agents. To define the environment infrastructure, an administrator has to perform the following steps:

1) to choose the roles her/his environment is going to support; often, this choice is implicit in the environment, as shown in the first example of the next section;
2) to define the policies by which the chosen roles can interact with each other or with the local resources.

This model of infrastructures leads to advantages at several stages of the application life cycle. At the *design stage*, roles permit the separation of concerns, which allows the designer to concentrate on the single (interaction) issue. At the *development stage*, the reuse of roles permits to avoid the implementation of common (already implemented by someone else) functionalities. At the *execution time*, more flexibility is achieved, since each environment can define its own local laws to rule interactions.

3.1 The Agent Level

Agents act on behalf of users and are in charge of carrying out tasks in an autonomous way. In this paper we disregard the internal constitution of agents, focusing on their external behaviors – captured by roles.

One of the characteristics of agents is *network-awareness*, which means that agents perceive the network not as a whole environment, but instead as a set of nodes, each one with given resources and services. No matter if agents actually move (along with their code and data) to different execution environments or not, each node is seen as a single, different environment, with its own resources and laws [4].

Nodes are thought as the places where agents interact, both with other agents and with resources, to carry out their task. Interactions among agents can involve agents of the same application, or can occur between agents of different applications. In the latter case, for instance agents can negotiate information or compete for limited resources.

Finally, agents are usually foreign with regard to a node, which means that they come from other nodes. Nevertheless, there could be agents local to a node, for instance representing services that nodes make available to incoming agents.

This scenario calls for the definition of general infrastructures, which can be adapted to different applications and situations.

3.2 The Infrastructure Level

In our proposal, an infrastructure is composed by a set of roles related to the same application context. Such definition implies two important features of an infrastructure:

- it is not bound to specific agents, which can belong to whatever applications, can have their own tasks and can be designed and implemented separately from the

site;

- it can host agents, providing a "wrapper" that not only accepts them, but also assigns them capabilities and a given behavior.

In Figure 1 the set of roles is represented by a "table" where each role is represented by a "hole", in which an agent can place itself in order to play such role. There could be several "holes" of the same role, if a site can host more than one agent playing the same role.

This idea of infrastructure enforces the locality concept introduced in the previous subsection. In fact, each site can decide how to organize the local hosting of agents and, by defining also mechanisms and policies, can rule the local interactions.

3.3 The Policy & Mechanism Level

This level deploys the policies that rule the local environment, and provides the mechanisms for the interactions both among agents and between agents and resources. While the previous level can be considered as the interface of a site toward the external world, this level enacts the site's laws.

The simplest example of policy is to allow or deny an interaction between two given roles. Thanks to their autonomy and reactivity, agents can handle situations where something is forbidden by local rules without giving up and aborting their job.

Though different from policies, we include mechanisms at this level because, as policies, they enable the interactions among roles and with local resources.

3.4 The Resource Level

At the last level we can find the resources local to a site. Usually, they are legacy resources that are hard to change or affect. So, it is important that the policy & mechanism level makes them available in a useful format for agents.

Also in this case, the use of roles helps in abstracting from the single agent or application, because mechanisms has to be enacted for a generic role, covering the wide range of actual agents that play such role.

Our proposal permits to disregard how local resources are managed, providing that appropriate access mechanisms are supplied.

4 Examples

This section presents some examples of applications where a role-based infrastructure is defined with the corresponding policies and mechanisms.

4.1 Auctions

The first example relates to the auctions. Auctions represent an interesting negotiation means in the Internet context. In an auction there are entities (called *sellers*) that make goods/resources available and entities (called *bidders*) that are interested in using/acquiring such goods/resources. Moreover, there are intermediate entities (called *auctioneers*) in charge of actually performing the negotiation. The price of the resources sold by sellers via an auction is not fixed, but it is dynamically determined by the interest of the bidders [1].

We can figure out that agents negotiate resources or goods via auctions, at given Internet sites representing auction houses [10]. Of course, the way the sellers, the bidders and the auctioneers interact is not bound to a given application or to a given environment, and so they can be considered roles that whatever agent can assume. In this case, the choice of the roles is tightly driven by the environment: the bidder, the seller and the auctioneer (see Figure 2). So the former step is accomplished.

The latter step relates to the choice of the local policies of interaction among roles and between roles and the environment resources.

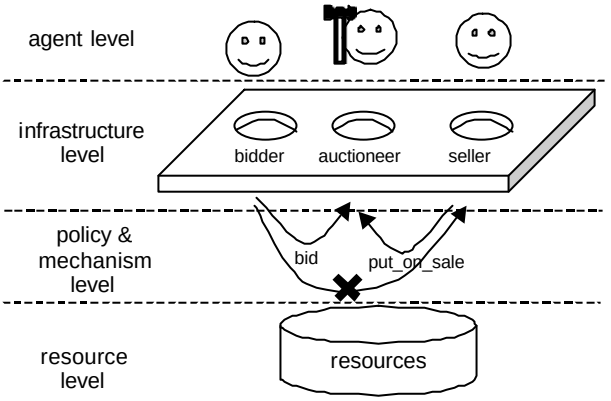

Figure 2. A role-based infrastructure for an auction house

The infrastructure built by roles is very flexible, because every environment can decide its own local policies and mechanisms that can be different from other sites' policies. There can be several reasons to have different policies or mechanisms. For example, in one environment the bidders could be allowed to talk each other, while in another site they cannot to avoid collusions; this permits to impose local rules or social conventions [11]. Another reason could be the different implementation of the auction mechanisms: the Figure 2

shows a message-passing oriented implementation, where, for instance, the bidder agent can bid by sending a message to the auctioneer agent. But if the implementation of the bidding mechanism is based on another model, the local policies must be different. For instance, if the auction relies on a data-oriented model such as tuple spaces [3], the bidding action is implemented as writing information in the local interaction space, as shown in Figure 3. This example shows that the same set of roles can be adapted to different implementations.

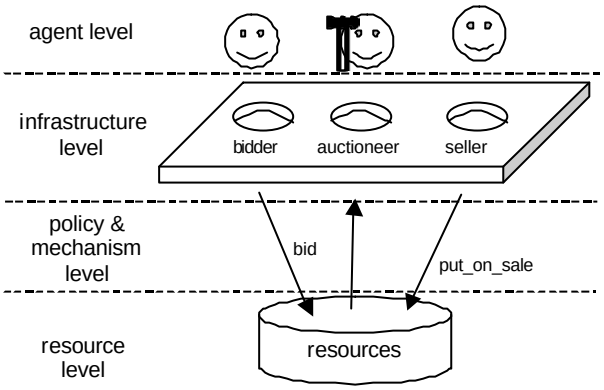

Figure 3. The same infrastructure relying on a data-oriented model

4.2 Restaurants

This example is taken from the human life, and may not be related to a real application based on agents; however, it is meaningful to understand how roles can be defined and how the interactions among them can be established.

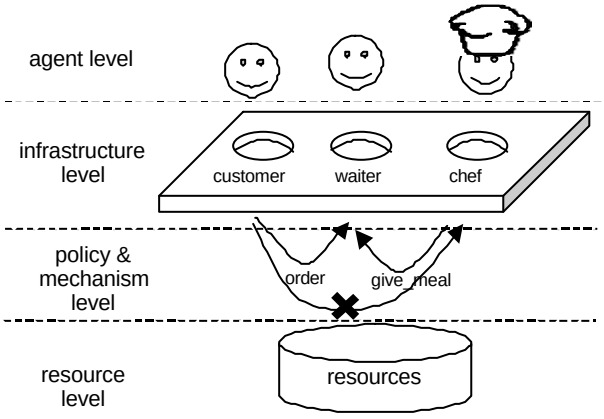

Figure 4. A role-based infrastructure for a restaurant

In this example, a node represents a single restaurant. We define three roles: the *customer*, the *waiter*, and the *chef* (see Figure 4). These roles can be thought as instances of the more general roles defined in a client-server model with an intermediate entity (the waiter) between the client (the customer) and the server (the chef), such as several 3-tier solutions.

The role of the customer can have the following capabilities: ask for the menu, order the meal, accept the meal, pay the bill. Note that "eat the meal" is not a capability of the role of customer, while it should be of the agent.

The waiter role has different capabilities: take order, order the meal (to the chef), accept the meal (from the chef), give the meal (to the customer), accept the payment.

Finally, the chef can: accept an order, give the meal. Again, the cooking of the meal is not an external capability of the chef *role*, but an intrinsic capacity of the chef *agent*.

The policy & mechanism level ensure that such interactions occur, for example that the customer order the meal to the waiter and the chef gives the cooked meal to the waiter. Some interactions may be disabled, such as the direct interaction between the customer and the chef (see Figure 4).

Now, let us suppose that the scenario changes. To save money, little restaurants do not have the waiter, but the chef itself is in charge of accepting and satisfying the customers' requests (see Figure 5).

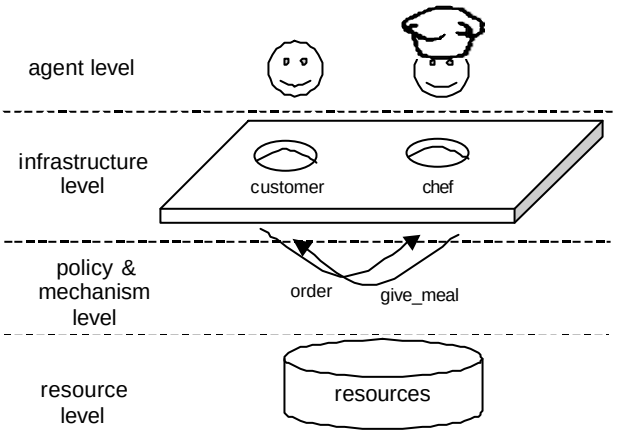

Figure 5. A restaurant without waiters

In this case, the infrastructure can exploit the same roles – disregarding of course the waiter role. On the one

hand, the agents can assume the same roles as in the previous scenario, and they can perform the same actions in the restaurant. On the other hand, the policy and mechanism level must be changed in order to allow the new interactions.

This example shows that the same roles can adopt different interaction models without affecting the role definitions and, as a consequence, the agents' way to achieve their goals.

5 Conclusions and Future Work

This paper has presented the building of infrastructures for agents, based on the concept of role. This permits to achieve separation of concerns between the interface of node to the agents (in terms of both capabilities and restrictions) and the actual implementation of policies and mechanisms, which must suit the local laws.

This way of building distributed infrastructures opens several research issues.

First of all, we are exploring the developing of a system for the definition of roles and their concrete exploitation in implemented applications. Such system should address interoperability to suit the openness of the Internet. We are planning to use XML for the definition of roles and XSL for the translation into documentation and real code.

Second, it could be interesting the availability of "repositories" of roles, from which agents can chose the more appropriate for their tasks. Which could be the most appropriate technology to create such repositories? And which access policies must be defined? If each repository is seen as a resource, meta-roles could be defined to rule the access to them. Moreover, the fact that agents could assume roles dynamically at runtime, imposes the study of methodologies to make this approach effective.

Third, effective tools are to be developed to support the building of infrastructures. They can help both the site developers and the site administrators, which can decide to change the local policies or mechanisms.

Acknowledgements

Work supported by the Italian MURST in the project "MUSIQUE – Infrastructure for QoS in Web Multimedia Services with Heterogeneous Access" and by the University of Modena and Reggio Emilia with a found for young researchers.

References

[1] Agorics, Inc., "Going, going, gone! A survey of auction types", http://www.agorics.com/new.html, 1996.

[2] M. Becht, T. Gurzki, J. Klarmann, M. Muscholl, "ROPE: Role Oriented Programming Environment for Multiagent Systems", *Proceedings of the 4th IFCIS Conference on Cooperative Information Systems (CoopIS'99)*, Edinburgh, Scotland, September 1999.

[3] G. Cabri, L. Leonardi, F. Zambonelli, "Auction-based Agent Negotiation via Programmable Tuple Spaces", *Proceedings of the 4th International Workshop on Cooperative Information Agents (CIA 2000)*, LNCS No. 1860, Boston (USA), July 2000.

[4] G. Cabri, L. Leonardi, F. Zambonelli, "Engineering Mobile-agent Applications via Context-dependent Coordination", *Proceedings of the 23rd International Conference on Software Engineering 2001 (ICSE)*, Toronto (C), May 2001, ACM Press.

[5] S. Clearwater, "Market-based Control: a Paradigm for Distributed Resource Allocation", *World Scientific*, 1995.

[6] N. R. Jennings, M. Wooldridge, eds., "Agent Technology: Foundations, Applications, and Markets", Springer-Verlag, March 1998.

[7] N. M. Karnik, A. R. Tripathi, "Design Issues in Mobile-Agent Programming Systems", *IEEE Concurrency*, Vol. 6, No. 3, pp. 52-61, July-September 1998.

[8] E. A. Kendall, "Role Modelling for Agent Systems Analysis, Design and Implementation", *IEEE Concurrency*, Vol. 8, No. 2, pp. 34-41, April-June 2000.

[9] B. B. Kristensen, K, Østerbye, "Roles: Conceptual Abstraction Theory & Practical Language Issues", *Special Issue of Theory and Practice of Object Systems on Subjectivity in Object-Oriented Systems*, Vol. 2, No. 3, pp. 143-160, 1996.

[10] T. Sandholm and Q. Huai, "*Nomad*: Mobile Agent System for an Internet-Based Auction House", *IEEE Internet Computing, Special issue on Agent Technology and the Internet*, Vol. 4, No. 2, pp. 80-86, March-April 2000.

[11] F. Zambonelli, N. R. Jennings, M. Wooldridge,

"Organizational Rules as an Abstraction for the Analysis and Design of Multi-agent Systems", *Journal of Software Engineering and Knowledge Engineering*, to appear, 2001.