Separation of Concerns in Agent Applications by Roles

Giacomo Cabri, Letizia Leonardi
Dipartimento di Ingegneria dell’Informazione
Università di Modena e Reggio Emilia
Via Vignolese, 905 – 41100 Modena – ITALY
{giacomo.cabri, letizia.leonardi}@unimo.it

Franco Zambonelli
Dip. di Scienze e Metodi dell’Ingegneria
Università di Modena e Reggio Emilia
Via Allegri 13 – 42100 Reggio Emilia – ITALY
franco.zambonelli@unimo.it

Abstract

In the development of agent applications, interactions are an important issue, which must be faced with appropriate methodologies and tools. A separation of concerns between the agents and their interaction needs is helpful in the designing and the implementation phases of the life cycle. In this paper we propose XRole, a system that helps in dealing with interactions. It is based on the definition of roles, which are intended as intermediaries between the application needs and the environment needs. XRole is realized exploiting the interesting features of the XML language. An application example shows the effectiveness of the approach.

1 Introduction

The Internet world will be more and more populated by software agents, which act on behalf of users to carry out tasks. In this scenario, the interactions among agents are becoming an important issue that must be taken into consideration at the different phases of the application development [4]. On the one hand, complex goals are faced by multi-agent systems by dividing the main goal in several simpler goals, each one assigned to one agent; this lets emerge the need for making agent of the same application interact for cooperating to carry out the global task. On the other hand, the Internet is becoming a place where resources and services for agents are available, but often agents have to compete to achieve them; this points out that interactions between agents of different applications are to be dealt with.

We propose to deal with agent interactions by a three-level model (see Figure 1), thought to manage both interactions among agents of the same application and interaction between agents of different applications. In this model, the application level is represented by the agents; the lowest level concerns the environment, which defines its own policies to rule agent-to-agent and agent-to-resources interactions. The middle level is the one focused by this paper, and concretely enables the separation of concerns between agents and their interaction needs. To achieve flexibility and dynamism, we propose to center it on the concept of role.

![Figure 1. Agent interaction model](image-url)

In the agent scenario, a role is defined as the behavior and the set of the capabilities expected for the agent that plays such role [8]. 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. Role definitions can be separated by agent definitions. 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 and then for different agents. Finally, roles are related to contexts, so that each environment can impose its own rules and can grant some local capabilities, forcing agents to assume specific roles. As mentioned before, roles represent behaviors that agents are expected to show; who expects such behavior are entities external to agents themselves, mainly organizations [12] and environments.

The contribution of this paper is to propose a system, called XRole, which enables the definition of roles for...
agents, in a way that is suitable to be exploited at different phases of the life cycle of an agent-based application. Such system is based on XML to exploit all advantages this language provides. Following the Aspect Oriented (AOP) philosophy [5], the aim of this system is the separation of concerns in the definition of the interactions among agents and support of their implementation. XRole separates the interaction-related behaviors and functionalities from the algorithmic parts of the agents [9].

Differently from pure AOP, XRole is thought to achieve not only performance and maintainability in software development; XRole adds flexibility in the definition and usage of aspects/roles, because it addresses the uncertainty and complexity issues of wide-open environments, such as the Internet.

2 The XRole Approach

2.1 Exploiting XML

XML [11] is a language for data representation that is becoming a standard for data interoperability in the Internet. XML represents data in a tagged form, and explicitly separates the treatment of data from its representation. This achieves both the well-appreciated feature of human-readability and the platform-independence required for the Internet. In addition, XML can be made capable of representing whatever kind of data and entity one is likely to find in the Internet: documents, services, as well as agents. These characteristics let us think that interoperability in the Internet will be information-oriented and based on XML, rather than service-oriented.

By using XML, the description of each role can be also presented to human people via an appropriate XSL sheet that transforms the information in a human-understandable document, such as a HTML page. This lets programmers develop their agents knowing which roles are available, for example by searching for appropriate roles in a repository.

Moreover, if the XML documents follow appropriate rules, they can be managed also by automated tools, and by the agents themselves, which can be used to understand the content of a XML documents and exploit or manipulate it without the need of the intervention of human people.

With regard to the catalogues of roles, we point out that XML carries great advantages in this context, since the same XML document can be exploited for both the description of a role and the operative use of it.

2.2 Definition of Roles

In XRole, roles are defined by XML documents that respect a well-defined DTD (see Figure 2).

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT role (name, description?, keyword?, action?)>
<!ELEMENT name (PCDATA)>
<!ELEMENT description (PCDATA)>
<!ELEMENT keyword (PCDATA)>
<!ELEMENT action (description, rules?, name?, parameter?)>
<!ELEMENT name (PCDATA)>
<!ELEMENT rules (PCDATA)>
<!ELEMENT parameter (name?, type?)>
<!ENTITY type (PCDATA)>
```

Figure 2. The DTD followed by XML roles defined in XRole

The XRole approach uses this kind of DTD to describe the role structure and the rules that define the role behavior. The DTD is based on XML elements which are recursively nested, each of them with an associated description.

```xml
<?xml version="1.0" encoding="UTF-8"?>
  <xsd:element name="role" type="xsd:complexType">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/>
      <xsd:element name="description" type="xsd:string"/>
      <xsd:element name="keyword" type="xsd:string"/>
      <xsd:element name="action" type="xsd:string"/>
    </xsd:sequence>
  </xsd:element>
</xsd:schema>
```

Figure 3. The XML Schema for XRole roles

The XRole approach uses this kind of DTD to describe the role structure and the rules that define the role behavior. The DTD is based on XML elements which are recursively nested, each of them with an associated description.

In alternative, a XML Schema can be used, such as the one reported in Figure 3.

By analyzing both the DTD and the XML Schema...
reported, one finds the main elements we have identified to describe a role. They are:

- **name.** The name given to the role. It would be useful that the name was unique, to identify a precise role all over the world. But we agree that it is a too strong requirement, so we envision that a name is valid in a given context (i.e., an application area, an Internet domain, and so on).
- **description.** A high-level description of the role. It is useful for designers to understand the aim of the roles, in human-readable sentences.
- **keyword.** Zero or more keywords can be used to identify the role. They are useful for human people, automated tools and also agents in the search of roles matching given criteria.
- **action.** It describes actions available to the agent if it assumes this role, in order to interact with other entities, such as other agents or execution environments. This description is quite general, specifying a name, a description, a return value, and a list of parameters.

The advantage of exploiting the XML language is that this definition of role can be extended, to meet specific requirements that will arise in the future.

### 2.3 Use of Roles

We emphasize that XRole is not bound to a given agent system or to a given interaction infrastructure. Instead, thanks to the high degree of interoperability provided by XML, it can be exploited at different phases and by different systems. In this section we show how XRole can be exploited at the different phases of the software production.

![Figure 4. An example of role defined in XRole](image)

**Figure 4. An example of role defined in XRole**

Design phase. The description of roles can be useful at the design phase, because gives the designers useful information about the agents that may compose the application under development. Like design patterns [2], the description does not provide code or libraries, but high-level suggestions that can be exploited at this stage of the software life cycle. In addition, sets of related roles can be defined, in order to provide a group of interacting roles to easily build up an application; in fact, each set can define not only the roles belonging to it, but also the relationships among the entities playing such roles. More than bare patterns, XRole descriptions can be automatically translated into actual code, as shown in the next subsection.

Thanks to the adaptability of XML, a role defined in XRole can be presented to the designers in different ways, to meet different requirements. The simplest way is to let an appropriate XSL sheet produce a HTML document that presents the information about the role. Figure 4 reports a simple role defined in XRole that will be assumed as an example in the following of the section.

Starting from the role defined by the XML document of Figure 4, the HTML document reported in Figure 5 can be created and published.

![Figure 5. A human-readable description of the XRole-example role](image)
In the agent world, an interesting negotiation means is the auction [10]. In an auction there are entities that make resources available and entities that are interested in using/acquiring such resources. The former ones are usually called sellers, while the latter ones are called bidders. Usually, there is an intermediate entity, called auctioneer, which actually performs the negotiation. The

Implementation phase. Using languages that support the notion of interface, such as Java, it can be very simple to derive an interface from a role defined in XRole. For example, by using the XSL document reported in Figure 6, the role defined in Figure 4 can be translated into the Java interface shown in Figure 7, where the name element of the role defined in XRole is used as the name of the Java interface.

```java
public interface MyRole {
    String description = "example of role";
    String keyword() = "example", "XRole";

    String talk_with(AGenID receiver, string message);
}
```

Figure 7. The XRole example translated into a Java interface

Moreover, situations where one agent plays different roles are usual. Also in this case, defining the roles in XML is very useful, because they can be combined by an appropriate XSL document.

Runtime. From the hosts' point of view, a hosting server that accepts agents using a given role must provide for the actual implementation of the actions defined in such role. For instance, referring to the role depicted in Figure 4, the server must somehow implement the action talk_with, that is, the talk_with method of the interface MyRole of Figure 7. However, XRole does not define the ways methods are implemented, and each architecture can choose the most appropriate one.

In a dynamic environment agents can search for an appropriate role by using a XML query language [6]. In addition, agents can dynamically search for appropriate roles that are needed, and assume the found roles at runtime. This is permitted by the fact that valid XML documents can be managed in an automated way, in this case by agents themselves, which can understand the description of a role decide whether assume or not such role, on the basis of different criteria.

3 Application Example

Figure 6. The XSL document to translate the XML role into a Java interface

More sophisticated views can be generated by appropriate XSL sheets, which tell the designers what they need. For example, a HTML documents can be created, which lists all roles that refer to the same keyword, or implement a given action.
price of the resources sold by sellers via an auction is not fixed, but it is dynamically determined by the interest of the bidders. The seller can set a reserve price, i.e., a price under which it does not want to sell the resource.

There are several different forms of auction, depending on the number of participants, on the criteria with which the resources are assigned, and so on. We focus on the auctions with one seller and multiple bidders at a time, ruled by several mechanisms: for example, English, Dutch, first-price and Vickery [1].

In the following we show how XRole can be exploited to describe roles that can be assumed by agents of auction applications. The reported XML documents are quite simple (due to the paper length limitation), but they aim at giving an idea of the features of XRole, and can be extended to be used in a real application.

The following role can be defined as “standard” by a set of auction sites, which either make the corresponding XML documents available to designers/developers or point to a “repository” from which roles can be retrieved. As stated in Subsection 0, these roles can be part of a set of roles related to auction applications, which defines also the relationships among roles. The implementations of the actions are then delegated to each site accepting agents that play these roles.

```xml
<role>
  <name>seller</name>
  <description>
    An agent attending an auction, which can bid to achieve a good on sale. The description is:
    <@action>
      <@description>Make a bid</description>
      <@value>bid</value>
      <d_name>bid</d_name>
      <@param>
        <par_name>good_id</par_name>
        <type>string</type>
      </@param>
      <@param>
        <par_name>price</par_name>
        <type>int</type>
      </@param>
    </@action>
  </description>
</role>
```

**Figure 8. A bidder role defined in XRole**

The first important role in an auction is the bidder. This role is assumed by an agent whose willing is to buy a resource on sale. In Figure 8 is reported the XML document conforming to XRole, which describes the role of the bidder.

In this example, two main actions are allowed for the agent that assumes such role: the bid, which is used to make a bid and the ask_status, which asks for the status of an auction.

The second significant role in the auction context is the seller. Figure 9 shown how a seller role can be described by XRole.

```xml
<role>
  <name>seller</name>
  <description>
    An agent attending an auction, puts a good on sale. The description is:
    <@action>
      <@description>Put a good on sale</description>
      <@value>good</value>
      <d_name>good_id</d_name>
      <@param>
        <par_name>good_id</par_name>
        <type>string</type>
      </@param>
    </@action>
  </description>
</role>
```

**Figure 9. A seller role defined in XRole**

In this example, an agent that assumes the role of seller is allowed (and also is expected) to put a good on sale by means of the put_on_sale action, specifying information about the good and the minimum price it wants to earn.

## 4 Other Role-based Approaches

The AOP has been exploited to implement the concept of role by E. Kendall [8]. She well describes the importance of modeling roles for agent systems. Our aim is to go beyond Kendall’s considerations, and to propose roles as intermediaries for the interactions between agent applications and environments. Our concept of role is more dynamic and aims at covering different stages of the life cycle of agent-oriented applications in a more practical way, concretely supporting both the design and the implementation phases.
AALAADIN [7] is a meta-model to define models of organizations. It is based on three core concepts: agent, group and role. Even if our approach is quite similar to the AALAADIN one, it differs for some reasons. First, we disregard the concept of group, while focusing on the interactions among agents and between agents and environments. Second, AALAADIN roles are tightly bound to the notion of agent, while our aim is to describe roles in a more independent way, both of applications and environments. Third, in AALAADIN environments are mainly modeled by service agents, which is generally acceptable, but do not cover all real situations, where also agents that play roles of “pure clients” must be taken into account.

The ROPE project [3] recognizes the importance of defining roles as first-class entities, which can be assumed dynamically by agents. It proposes an integrate environment, called ROPE (Role Oriented Programming Environment), which can be exploit to develop applications composed by several cooperating agents. Rather than defining an integrated but close environment, XRole aims at proposing an open methodology to define agent roles. It addresses interoperability and also the dynamic use of roles. Moreover, XRole can be used to define roles also for interactions among agents that do not belong to the same application (i.e., are competitive): this is a relevant aspect in the design of applications for wide-open environments, such as the Internet.

5 Conclusions and Future Work

This paper has presented XRole, an XML-based system designed to define roles for agent applications. This system can be useful exploited to enable separation of concerns at different phases of the life cycle of applications. The XML language gives flexibility and interoperability to system.

Some research directions for future work may be the following.

First, XRole is a work in progress, and some issues related to the actual implementation of roles are to be faced. The fact that an agent can assume a role at runtime must be carefully evaluated, and the implementation may require appropriate mechanisms and constructs, possibly provided by the implementation language.

Second, it could be interesting the definition 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 can assume roles dynamically at runtime, imposes to resolve the issues sketched in the previous paragraph, to make effective an approach based on role repositories.

Finally, but we are going to test XRole in applications different from the one proposed in this paper, to verify its applicability in a wide range of application areas.

Acknowledgements. Work supported by the NOKIA Research Center Boston, by the Italian MURST within the project “MUSIQUE” and by the Italian Research Council (CNR).

References