Multimed Tools Appl (2015) 74:4561–4582
DOI 10.1007/s11042-013-1822-0

Developing a virtual trade fair using an agent-oriented approach

Inmaculada Remolar · Alejandro Garcés · Cristina Rebollo · Miguel Chover · Ricardo Quirós · Jesús Gumbau

Published online: 17 December 2013

Abstract Online virtual trade fairs are becoming a popular way of establishing economic trade relationships nowadays. In the current context of the world economy, new ways of creating virtual environments where suppliers can show their products to potential customers are increasingly in demand. Some people can be found in this kind of scenario representing a role such as, for example, the administrator, the customers or the suppliers. The interaction between them is one of the most important characteristics of these commercial environments. Taking all these concepts into account, this article presents a 3D virtual trade fair that has been defined using an agent-oriented approach. This approach makes it possible to model, understand and implement the virtual economic environment, where interaction is the norm. Moreover, it makes it easy to add new components and meet new requirements in order to develop an open architecture that continuously changes and evolves. In this virtual trade fair, the chosen methodology has been Gaia, widely used for agent-oriented analysis and design of multi-agent systems.

Keywords Multi-agent systems · Business environment · Virtual trade fairs · Virtual worlds

I. Remolar (✉) · A. Garcés · C. Rebollo · M. Chover · R. Quirós · J. Gumbau
Institute of New Imaging Technologies, Universitat Jaume I, 12006 Castellón, Spain
e-mail: remolar@uji.es

A. Garcés
e-mail: agarces@uji.es

C. Rebollo
e-mail: rebollo@uji.es

M. Chover
e-mail: chover@uji.es

R. Quirós
e-mail: quiros@uji.es

J. Gumbau
e-mail: jgumbau@uji.es

1 Introduction

In the 21st century, it is very important to find new ways of establishing commercial relationships by minimizing economical resources. Nowadays, the Internet is opening new virtual markets that make it easy to develop virtual business environments. They are cheaper than real trade fairs and easier to plan. Many people are trying to make money in virtual worlds, because of the current difficulties of doing so in the real world [24]. In these virtual worlds, exhibiting companies can show their products to potential customers around the world without having to visit them personally [5]. International participants can "attend" with little or no expense. This helps companies save and earn money, and is thus the main reason they have become so popular.

Technological companies that develop virtual-meetings software, such as Unisfair [23], ON24 [15] and Second Life [19] say that they are seeing a big increase in demand. In this context, the number of virtual trade fairs available on the web is increasing every day [13] [3] [12] [28] [18]. However, software architectures that contain many dynamically interacting components and complex coordination protocols are typically difficult to specify and implement [25]. Most of them have been defined using an object-oriented methodology. Nevertheless, this type of methodologies consider tangible things, roles organizations, events and even interactions as candidate objects, whereas these need to be clearly distinguished and treated differently in a typical interaction system [20][27]. This has in turn led to the search for new computational abstractions, models and tools that allows software engineers to easily conceptualize and implement interacting systems. Software agents and multi-agents systems can perform this task successfully [20].

The main reason for performing an agent-oriented approach of the virtual environment is that the concept of an agent as an autonomous system, capable of interacting with others agents, is a natural concept for software designers [26] [10]. Some virtual environments have been defined used this approach [25]. Vosinakis and Panayiotopoulos present in [25] a tool for the construction of virtual worlds with autonomous entities targeted for a specific group of applications, such as simple simulation systems, educational applications or multimedia presentations.

The first thing that has to be performed in the design of a virtual environment is to identify the agents. An agent is an abstraction of a component in the system that has some properties. The first one is the *autonomy*, i.e., the agent can make decisions about what to do based on a determined state. He has also *reactivity*, i.e., he is situated in an environment and is able to perceive it and to respond to changes that occur in it. Moreover, an agent has *pro-activeness*, i.e., he does not simply act in response to the environment; he is able to take the initiative. Finally, he has *social ability*, i.e., agents can interact with other agents in order to achieve his goals.

Regarding the virtual trade fair, many people are involved in this 3D business environment [16]. The main role is the one represented by the *exhibiting companies*. They have a stand in a pavilion and show their products to the visitors. Other types of users are the customers or *visiting companies*. They can walk around the trade fair and visit the different pavilions where the suppliers provide some information about the products they sell. This makes it possible to find and compare the products, so the visitors are able to select the most appropriate products from among those of various exhibiting companies. Finally, as in real life, every trade fair has an *administrator*. Here the administrator is in charge of organizing the virtual world and assigning permissions to the other users: exhibitor and visitor users.

In order to perform an agent-oriented approach of this business-to-business environment, its analysis and design has been performed using Gaia [27] [29] [4]. Others methodologies have been studied [9] but the Gaia concept is considered as a standard in the analysis of the multi-agent systems. Gaia is an agent-oriented methodology, which uses a role-based organizational metaphor. It is founded on the concept of a multi-agent system as a computational organization consisting of various interacting roles. When applying Gaia, the analyst increasingly moves from abstract to concrete concepts: analysis and design can be thought of as a process of developing increasingly detailed models of the system to be constructed.

A virtual trade fair has been developed based on this approach [11]. Regarding the implementation process, the 3D environment has been created using one of the latest technologies in video games: Adobe Director (Fig. 1). This software, together with a free plug-in, makes it possible to create and publish interactive games for the web on Mac and Windows desktops. It also enables the designer to virtually integrate any major file format, including FLV and native 3D content.

The rest of this article is organized as follows. The planning of the agent-oriented analysis and design of the virtual fair is detailed in Section 2. Starting from the beginning, the virtual environment of the trade fair is analyzed in Section 3. Then, the specification of the agent-oriented system is considered in Section 4. Once this initial process has been detailed, the design of the virtual trade fair is explained in Section 5 and its implementation is overviewed in Section 6. Section 7 discusses an use case performed with the developed approach and finally, conclusions and future work are presented in Section 8.

2 Agent-oriented approach of the virtual fair

In order to provide an agent-oriented approach, the architecture of this virtual world is classified in two parts: the set of agents that play a role in the virtual trade fair and the physical environment that the agents can interact with. Following this classification, the system's definition has been divided in two main parts (Fig. 2).

In the first step, the model of the environment is specified. Gaia does not give information about the most appropriate methodology to perform this specification. Then, Object-Z language [21] [7] has been chosen for this purpose. In order to specify the social environment, some classes have been defined with their consistency constraints. Moreover, some computational objects have also been created in order to be used by the agents as global information resources.

Secondly, once the social environment has been defined, an analysis and design of the Multi Agent-System (MAS) in this virtual environment is provided. This step has been performed by following the Gaia methodology [27] [29] [4].

Fig. 1 An aerial view of the outside scenario in the virtual trade fair

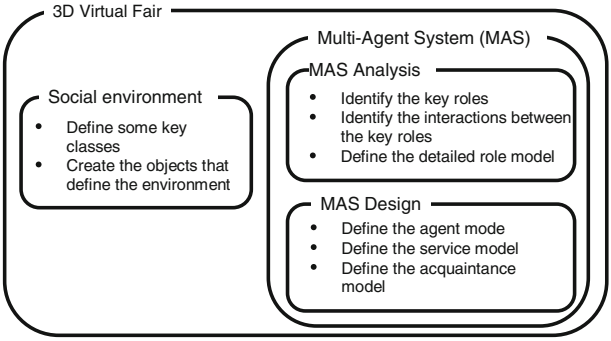

Fig. 2 A scheme of the steps to develop to perform an agent-oriented approach of the virtual fair

Regarding the analysis of the MAS in the virtual trade fair, some tasks have been developed:

- The roles in the system have been identified. This process outputs a list of roles in the virtual world, providing an informal description of each of them.
- The associated protocols have to be identified and documented for every identified role. Moreover, the different interaction patterns between the roles that appear in the system also have to be documented. This task outputs an interaction model.
- The third task performs the detailed role model. This includes the key roles, detailing their permissions and responsibilities, together with the protocols and activities in which they participate.

Once the analysis of the system has been performed, the design of the MAS has to be faced. The tasks that have been carried out for this purpose are explained below.

- Firstly, the agent model has been developed. It determines the number of agents or avatars that are in the virtual world representing each one of the roles.
- Secondly, the service model has been defined according to the activities, protocols, and the active and safe properties. This model specifies the services developed by every one of the agents in the virtual world.
- Finally, the model of collaborators (acquaintance model) has been built from the interaction and agent models. It determines the communication links that exist between the different agent types.

Once the environment model and the MAS has been analyzed and designed, the implementation of the virtual trade fair can be carried out. The next sections explain these processes in detail.

3 Environment specification

In order to start the modeling of the virtual trade fair (Fig. 3), a specification of the physical environment in this virtual world is required. This process is performed using the Object-Z specification language [21] [7]. This language allows us to determine the main classes of objects found in these kinds of virtual scenarios. Finally, the social objects that appear in this virtual world are defined in order to provide some information of the environment of the agents.

Fig. 3 A view of the outside scenario where some pavilions can be observed

3.1 Predefined types

In the object class definitions, some predefined types are used. Some of them are the usual types used in a language programming. Nevertheless, some special types have been defined according to the requirements of the virtual trade fair design. These new predefined types are:

- MULTIMEDIA type: it defines the multimedia information that can be uploaded by every exhibiting company in its stand.
- COM_INFO type: this type deals with the basic information of the exhibiting companies.
- VIEW type: it defines the views or images of the world that can be assigned to every user.
- POSITION type: this defines the 3D position in the virtual world. For example, it can be used to position a company agent in the virtual trade fair.

3.2 Object classes

Some object classes have been defined in the virtual environment. They are detailed in this section. In every definition, some processes have been expanded, such as *Init*, *Insert*, etc. However, others processes and functions are not included for reasons of brevity.

The main object class is the VIRTUAL_TRADE_FAIR (Fig. 4). It defines the whole virtual world. Two parameters are required in order to build an object of this class. The first one establishes the maximum number of pavilions that are in it (*n1*). Pavilions are the main buildings in a virtual trade fair. They are buildings where the exhibiting companies set up their stands. The other required parameter (*n2*) establishes the maximum number of thematic routes that a user can visit. The elements included in the virtual fair are defined in this object class, such as the *pavilions*, the *routes* (an established path from a stand to another of different companies that market products of the same topic), the *visitors*, users that want to visit the virtual world and the data that every company can show to the potential customers, *datac*. Three processes have been detailed in this case: *INIT*, *InsertDATACOM* and *InsertPAVILION*.

Another basic object class in this virtual world is the PAVILION class (Fig. 5). Only one parameter is required to create an object of this class: the maximum number of stands that are going to be allocated in this building. Some properties have been defined in this class: the identifier, *idPavilion*, the *name*, a flag that indicates if this pavilion is *active* or not and, finally, the objects of the class STAND (*stands*) In this class, two processes have been detailed: *INIT* and *InsertSTAND*.

The exhibiting companies that are interested in selling their products have a stand in the pavilion. This key object is defined in the class STAND (Fig. 6). Some multimedia material can be uploaded in this objects, so that the companies can show their catalogs or promoting videos to the visiting people in the virtual fair. Then, in order to create an object of this class,

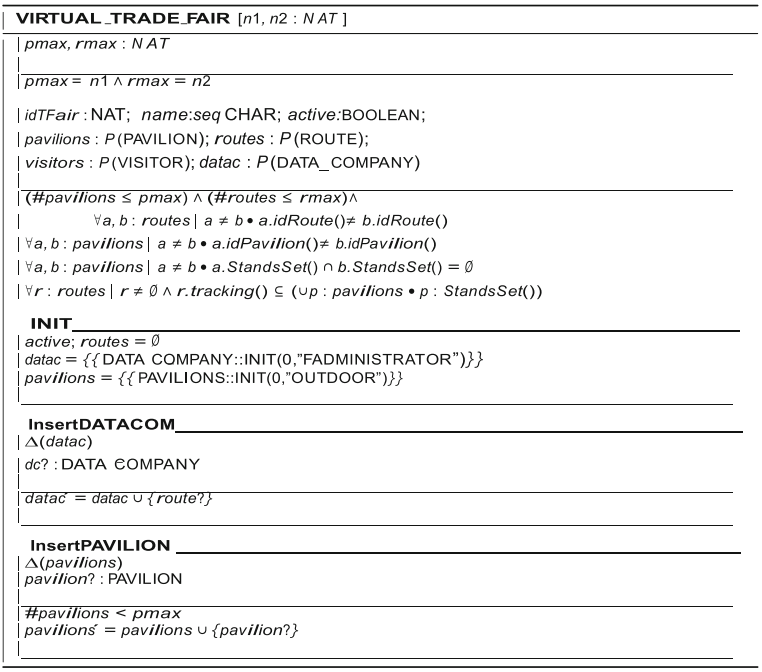

Fig. 4 The object class VIRTUAL_TRADE_FAIR

the number of maximum multimedia materials is required (*n*). Every stand is identified by an identifier, *idStand*, and the name, *name*. The company that owns it is identified by *idExhibitor*. The other attributes are related with the multimedia material: *image* and *mms*.

The object class ROUTE (Fig. 7) allows the visitor to move around the virtual trade fair following a thematic route from one stand to another. Different companies that market similar

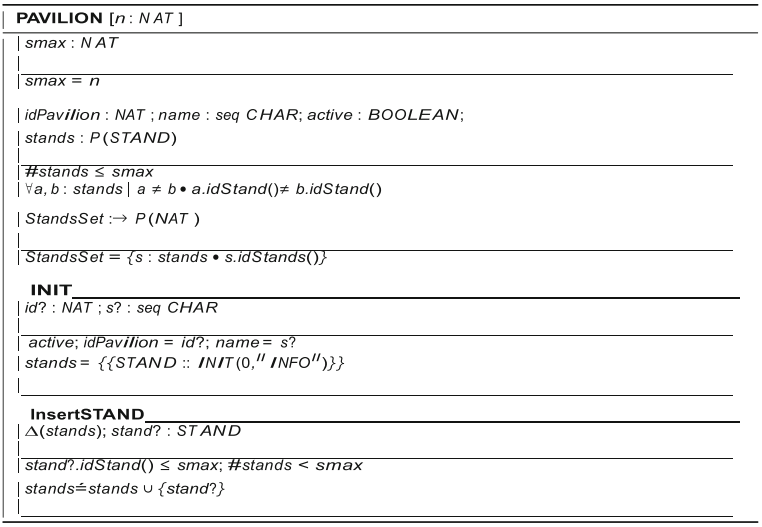

Fig. 5 The object class PAVILION

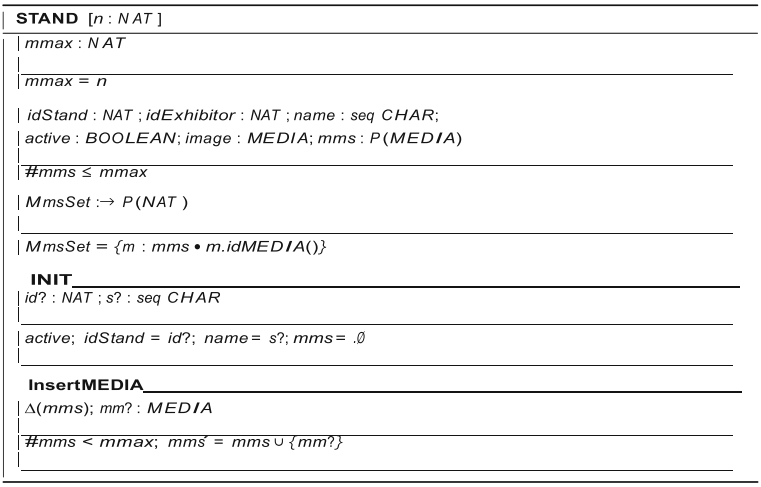

Fig. 6 The object class STAND

products can be visited following this precompiled path. If the visiting company chooses this way to tour the trade fair, he/she will be able to skip directly from one stand to the next in the route. This class requires the maximum number of stands (n) that are stored in the path in order to build an object. The properties that define these objects are the identifiers (*idRoute*), the *name*, if they are *active* and the final sequence of company identifiers, *tracking*.

Finally, other classes have been defined but they are not explained in this section for reasons of brevity. This is the case of VISITOR, MEDIA and DATA_COMPANY classes. The VISITOR class represents the data that a user has to provide in order to be allowed to visit the virtual fair. The MEDIA class represents the multimedia material that an exhibiting company can show in its stand to potential customers. All of the basic information that a company has to provide to register itself in this virtual world are defined in the DATA_COMPANY class.

3.3 Social objects

Once the object classes have been defined, some social objects are created in order to make the general information of the virtual world accessible to the agents. The most important ones are listed below, organized by their types.

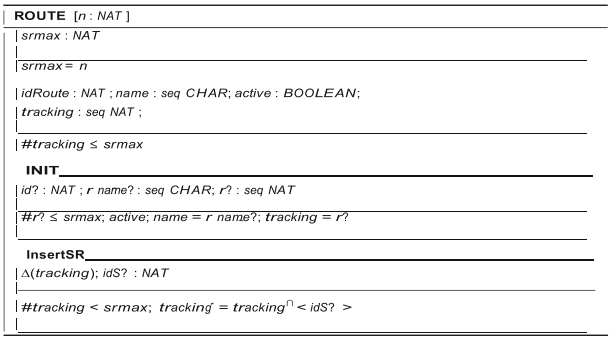

Fig. 7 The object class ROUTE

- *VIRTUAL_TRADE_FAIR* type: the object *VirtualFair* represents the virtual trade fair. It describes its content and functionality. It also defines all the low-level information for the management of the virtual world, such as the name of the pavilions, the stands and the routes. It also gives the agents access to the basic and multimedia information of the exhibiting companies.
- *COM_INFO* type: some objects have been created in order to allow the different agents to exchange some commercial information, such as the object *companyInfo*.
- *MULTIMEDIA* type: this type of objects allows the users to exchange and transmit multimedia elements, such as the object *MMelements*.
- *BOOLEAN* type: other relevant objects have been created for dealing with the authorization or negotiation processes. These objects check some of the permissions of the user and, if they are evaluated as TRUE, some actions are allowed for this user. For instance, *registrationOK* allows the user to be registered in the virtual world, *publishOK* to publish some multimedia material, *standOK* to have a stand in a determined pavilion, *tourOK* to follow a path or route, *commuteWorldOK* to change the scenario the user is in and also allows the user to pass to another scenario through a door access and *moveOK* to walk around the virtual world freely.
- *VIEW* type: the created objects of this type contain the views that the user can have in the world, according to the authorization previously obtained by the fair administrator. For example, view stores the current view, *tourView* keeps the view of the tour, *commuteView* stores the view that the user will have once he/she has passed through one virtual door to another scenario and, finally, *moveView* contains the view of the user when he/she moves freely around.
- *POSITION* type: this type contains the coordinates and orientation of a particular object. In this case, two objects have been defined: *companyPos*, that defines the current position and *positionAfterCrossing*, that establishes the position after passing through a door to another virtual scenario.

4 Analysis of the multi-agent oriented system

Gaia is an agent-oriented methodology, which uses a role-based organizational metaphor. The virtual trade fair is defined as a computational organization consisting of various interacting roles. Then, in order to perform an appropriate analysis of the virtual world, the key roles played by the users in the system are identified and the interaction between them is also detected. Finally, the key roles are formally defined.

4.1 Key roles in the system

In the virtual world, four key roles have been identified:

- *Company* (*CO*) which describes the individual companies that participate in the virtual trade fair. These companies can be exhibitors, with a stand in a pavilion, or simply visitors of the virtual world, acting as potential customers.
- *Fair Administrator* (*FA*) is the administrator of the virtual world. This role is the one that makes the relevant decisions and allows the users to perform some actions.
- *WorldCreator* (*WCr*) is responsible for managing the visual content of the world, from the user's perspective.

– *GateTrigger* (*GT*) represents some doors or gates that allow the user to change the virtual scenario he/she is in.

4.2 Interaction model

Some interactions have been detailed in this interaction model. They document the communication between the key roles documented in the previous sub-section. The parameters that require or return these processes have been previously detailed in the section 3.3.

The first process to be performed by the companies that want to visit the virtual trade fair is to ask for registration from the fair administrator, *RegistrationRequest*. Based on this, the first interaction detected occurs between the *Company* (CO) and the *FairAdministrator* (FA) roles. In this process, the company provides the *companyInfo*. This interaction produces other process, called *RegistrationResponse* that provides the answer of the FA role. The information involved in this process is *companyPos*, the position of the company in the virtual world, and *registrationOK*, that provides information about the result of the process. This interaction is illustrated in Fig. 8(a).

If a company is interested in uploading some multimedia material in its stand, this user has to ask for permission from the fair administrator. This interaction has been called *MMPublishRequest* and it is illustrated in Fig. 8(b). The CO gives the FA the *companyInfo* and the *MMelements*. FA creates a *companyRecord*, uploads all of the multimedia material in the stand and gives an answer to the CO, *publishOK*.

Another interaction is detected when a company is interested in visiting the virtual trade fair, *TourRequest*, documented in Fig. 9(a). This interaction involves three roles, the *Company* (CO), the *FairAdministrator* (FA) and the *WorldCreator* (WCr) roles. The company CO asks the FA for permission. The FA checks the data of the company, and gives an answer, *TourResponse*. If everything is OK, the role WCr creates the appropriate scenarios for the user (*PersonalizedWorldRequest*) and, finally, informs the CO that can start the free tour (*PersonalizedWorldResponse*).

If the company is interested in a free tour, its user should be able to change of scenery. For instance, this user can be in the outdoor environment and be interested in visiting a determined pavilion. This process, called *CrossDoorRequest* and graphically described in Fig. 9(b), involves again three roles: the *Company* (CO), the *GateTrigger* (GT) and the *WorldCreator* (WCr) roles. The *Company* asks to the GT for changing the scenario he/she is in. The *GateTrig*ger asks for the creation of a new scenario to

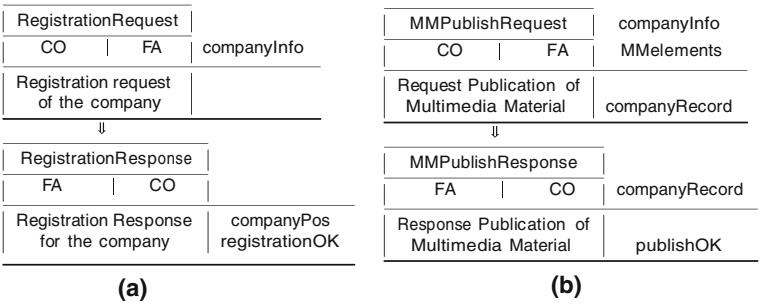

Fig. 8 **a** Asking for registration and **b** asking for uploading multimedia material

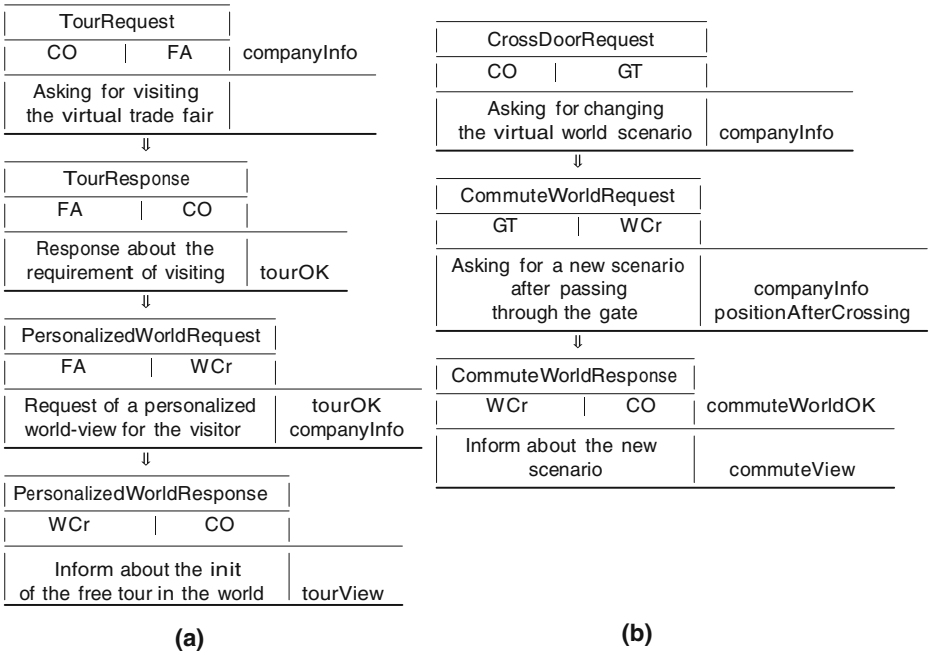

Fig. 9 **a** Asking for a tour visit and **b** asking for another scenario

the *WorldCreator* (*CommuteWorldRequest*). This one creates the new scenario and informs about it to the company user (*CommuteWorldRerponse*).

If a company is interested in exhibiting its products, it has to ask for a stand in a pavilion (*StandRequest*). Then an interaction is also produced between two roles: the *Company* (CO) and the *FairAdministrator* (FA). CO asks the FA role for a stand. As a response (*StandResponse*), FA gives the company permission (*standOK*) and the position of the stand in the world (*companyPos*). This is illustrated in Fig. 10(a).

If the company that visits the virtual fair is following a route, this user can skip from one stand to another (*MoveToRequest*). In each stand, the visitor can access the multimedia material uploaded in the stand. In this process, the involved roles are the *Company* (CO) and the *WorldCreator* (WC). The interaction is graphically detailed in Fig. 10(b).

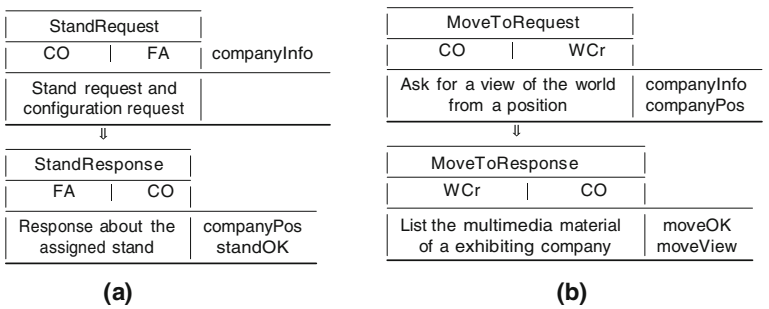

Fig. 10 **a** Asking for a stand and **b** asking for a movement

4.3 The role model

Once the key roles and their interactions have been identified, the Gaia methodology requires a formal definition of each role. In this step, four attributes have to be defined for every role: *protocols*, that define the way that this role can interact with others *activities*, actions that do not need another role in order to be carried out, *permissions*, the "rights" associated with the role, and *responsibilities*.

This last attribute determines its functionality and it is divided into two types: *liveness* properties and *safety* properties. Liveness properties describe those states of affairs that an agent must bring about, given certain environmental conditions. In contrast, safety properties are invariants that are kept while executing.

Figure 11 details the *Company* role definition. In this schema, the different actions that a company can represent are detailed. After the registration process, a company can be an *Exhibitor* or make a *Tour*. If he/she chooses to configure the stand, he/she has to develop the steps detailed in the *Exhibitor liveness* properties. Nevertheless, if he/she is interested in moving around the virtual trade fair, he/she has to develop the actions detailed in the *Tour liveness* properties.

The role *FairAdministrator* is documented in the Fig. 12. This role is in charge of managing the virtual trade fair. Then, this role has to detail the processes developed in order to perform some actions, called in the role schema *MakeRegistration*, *NegociateTour*, *AssignStand*, *PublishMultimedia*, *SetChat*.

Figure 13 detail the role of the *WorldCreator*. This role manages the final render of the virtual world, depending on the position of every user. It performs all the actions in order to change the view from one to another point of view in the scene.

The *GateTrigger* role (Fig. 14) manages the request for changing the scenario a user is in. If the demand is accepted, is the role in charge of asking for a change of scenario to the *WorldCreator* role.

5 Design of the agent-oriented approach

The design of a system following Gaia is oriented to transform the analysis models into a sufficiently low level of abstraction so that traditional design techniques (including object-oriented techniques) may be applied in order to implement agents [26].

The Gaia design includes three models: the *agent model*, which defines the classes and instances of agents to be used in the system, the *services model* describing the main services required to perform the role of agent, and the *acquaintance model*, that describes the lines of communication between the different agents in the system.

5.1 Agent model

In the agent model, a class of roles contains a small number of agents or, in most cases, a single agent. The instances of these classes of roles are documented using an annotation system shown in Table 1 (reproduced from [27]).

Finally, the agent model designed for the Virtual Trade Fair is shown in Fig. 15. One agent can represent the role of the *FairAdministrator* in the virtual fair. The same restriction is done for the *WorldCreator* role: only one agent can be created that represents this role. Regarding the *Company* role, none or more than one agent can be present in this virtual world and the same happens with the *GateTrigger* role: there can be none or more than one virtual door to other scenarios.

ROLE SCHEMA **Company**

DESCRIPTION
Organization that requires at least one of the two following behaviors within the fair: to be a visitor or to be an exhibitor with an assigned stand.

PROTOCOLS & ACTIVITIES
RegistrationRequest, AwaitRegistrationResponse, StandRequest, AwaitStandResponse, ChatRequest, RespondToCallers, TourRequest,AwaitTourResponse, AwaitPersonalizedWorldResponse, SetMovement, MoveToRequest, ShowView, ListStandsRequest, SelectStand, CrossThresholdRequest. AwaitCommuteWorldResponse. AwaitListStandsResponse, AwaitCommuteWorldResponse. ProduceMultimedia, MMPublishRequest. AwaitMMPublishResponse

PERMISSIONS

reads	tourOK, listStands, commuteWorldOK, moveOK tourView, commuteView, moveView
generates	companyInfo, MMelements, view
changes	companyPos

RESPONSIBILITIES

Liveness

Company = RegistrationRequest. AwaitRegistrationResponse. (**Exhibitor** | **Tour**)∗

Exhibitor = [StandRequest. AwaitStandResponse]. (**MultimediaUpdate** | ChatRequest | RespondToCallers)∗

Tour= TourRequest. AwaitTourResponse. AwaitPersonalizedWorldResponse. **Move**

Move =(SetMovement . MoveToRequest. ShowView | ListStandsRequest. AwaitListStandsResponse. (SelectStand. MoveToRequest, ShowView)+ | CrossThresholdRequest. AwaitCommuteWorldResponse. ShowView)∗

MultimediaUpdate = ProduceMultimedia. MMPublishRequest. AwaitMMPublishResponse

Safety
- (tourOK ∧ commuteWorldOK) ∨ (tourOK ∧ moveOK) ∨ (commuteWorldOK ∧ moveOK)
- tourOK ∧ tourView ≠ nil ⇒ view = tourView
- commuteWorldOK ∧ commuteView ≠ nil ⇒ view = commuteView
- moveOK ∧ moveView ≠ nil ⇒ view = moveView

Fig. 11 *Company* role definition

5.2 The services model

Once the number of agents representing each identified role has been established, the following step is to identify the services associated with them, and to specify the main properties of these services [27]. The services that an agent will perform are derived from the list of protocols, activities, responsibilities and the liveness properties of the role. For each service that may be performed by an agent, it is necessary to document its properties. Specially, we must identify the *inputs*, *outputs*, *pre-conditions*, and *post-conditions* of each service.

Due to reasons of brevity, in this paper only is detailed the service model designed for the *FairAdministratorAgent* (Table 2).

ROLE SCHEMA **FairAdministrator**

DESCRIPTION
Manages both the general information of the organizations and their custom items for the virtual world

PROTOCOLS & ACTIVITIES
AwaitRegistrationRequest, SetRegistration, RegistrationResponse, AwaitTourRequest, CheckTour, TourResponse, PersonalizedWorldRequest, AwaitListStandRequest, CreateListStands, ListStandsResponse, AwaitStandRequest, SetStand, StandResponse, AwaitMMPublishRequest, CheckMM, MMPublishResponse, AwaitChatRequest, CreateChatSession

PERMISSIONS
reads companyInfo, MMelements
generates virtualFair, registrationOK, publishOK, tourOK, standOK, companyPos

RESPONSIBILITIES
Liveness
FairAdministrator = (**MakeRegistration** | **NegotiateTour** | **AssignStand** | **PublishMultimedia** | **SetChat**)*
MakeRegistration= AwaitRegistrationRequest. SetRegistration. RegistrationResponse
NegotiateTour= AwaitTourRequest. CheckTour. TourResponse. PersonalizedWorldRequest. [**GuidedTour**]
GuidedTour = AwaitListStandsRequest.CreateListStands. ListStandsResponse
AssignStand= AwaitStandRequest. SetStand. StandResponse
PublishMultimedia= AwaitMMPublishRequest. CheckMM. MMPublishResponse
SetChat = AwaitChatRequest.CreateChatSession
Safety
¬ companyRegistration(companyInfo) ⇒
publishOK = false ∧ tourOK = false ∧ standOK = false

Fig. 12 *FairAdministrator* role definition

The *Registration* service is related to the *MakeRegistration* property. It requires as input from the *companyInfo* data. If there is no problem, this services returns the position of the stand in the pavilion, *companyPos*, and a flag indicating if the registration process has been successful, *registrationOK* .

The *GuidedTour* service, related to the *GuidedTour* property, requires the *companyInfo* as input and a list of stands, *listStands*, as outputs. In order to follow a guided tour in the virtual world, the company has to be previously registered. After the service has been performed, if an agent requires a tour (*tourOK=true*), the list of stands will not be empty (*listStands≠Ø*)

The rest of services, related to the *NegotiateTour*, *AssignStand*, *PublishMultimedia* and *SetChat* properties, are not explained in the text because of its simplicity.

5.3 The acquaintance model

This last model defines the communication links that exist between the different agent types. Figure 16 shows the ones defined in this virtual world. The *FairAdministrator* and *GateTrigger* agents can establish communication with the *WorldCreator* and the *Company*

ROLE SCHEMA **WorldCreator**

DESCRIPTION
Receives requests from all individuals in the system
in order to create the virtual world view according user perspective

PROTOCOLS & ACTIVITIES
AwaitPersonalizedWorldRequest, CreateView,
PersonalizedWorldResponse, AwaitCommuteWorldRequest,
CommuteWorldResponse, AwaitMoveToRequest, MoveToResponse

PERMISSIONS
reads virtualFair, companyInfo, companyPosition,
companyOrientation, positionAfterCrossing,
orientationAfterCrossing tourOk, commuteWorldOK
generates tourView, commuteView, moveView, moveOK

RESPONSIBILITIES
Liveness
WorldCreator = (AwaitPersonalizedWorldRequest. CreateView.
PersonalizedWorldResponse | AwaitCommuteWorldRequest.
CreateView. CommuteWorldResponse |
Await MoveToRequest. CreateView. MoveToResponse)*
Safety
- tourOK = false ⟹ tourView = nil
- commuteWorldOK = false ⟹ commuteView = nil
- ¬ worldAvailable(companyPos) ⟹ moveOK = false ∧
moveView = nil

Fig. 13 *WorldCreator* role definition

ROLE SCHEMA **GateTrigger**

DESCRIPTION
Receives requests from the companies in order to cross an access gate
and get into another virtual area of the fair

PROTOCOLS & ACTIVITIES
AwaitCrossThresholdRequest . CheckCrossing. CommuteWorldRequest

PERMISSIONS
reads companyInfo
generates commuteWorldOK, positionAfterCrossing
orientationAfterCrossing

RESPONSIBILITIES
Liveness
GateTrigger = (AwaitCrossThresholdRequest. CheckCrossing.
CommuteWorldRequest)*
Safety
¬ infoAvailable(companyInfo) ⟹ commuteWorldOK = false

Fig. 14 *GateTrigger* role definition

Table 1 Instance qualifiers

Qualifier	Meaning
n	there will be exactly n instances
m..n	there will be between *m* and *n* instances
*	there will be 0 or more instances
+	there will be 1 or more instances

agents. However, the *WorldCreator* and the *Company* agents can communicate with all the agents in the virtual world, regardless of the role they play.

6 Implementation details

Several technologies have been employed in the implementation of the virtual fair. Firstly, *3D Studio Max* has been used to model all the objects. The objects have been modeled taking into account tight requirements of real-time applications that make it possible real-time performance. *Google SketchUp* has also been employed because of its simplicity. In order to obtain interactivity, only one object of each type has been built and instanced as many times as it has been needed.

Textures are dynamically downloaded from the data server. The website address for these textures can be obtained from the *XML* document, previously generated. They are assigned to the instanced objects in order to perform the final configuration. This download is performed by streaming, so this process is transparent to the user. The appearance of the virtual fair can be changed just by changing the textures. Thus, it is very easy to vary the decoration of the scenarios.

The most important factors are the polygon count and the size and amount of the used textures. All the avatars have been modeled with 2.380 polygons to allow a high number of characters to be visible on the screen. These characters use a texture size of 512×512 pixels, which is enough to offer a good quality appearance at a medium-close distance. The total amount of polygons for the outdoor scenario of the fair with 6 pavilions is 21.720 polygons. The number of polygons used in representing a pavilion with 36 visible stands, is 9.692. The total amount of Kbytes for the fair is 1.608 KB and 2.581 KB for each character with animations.

Once the objects have been modeled, the virtual world has been implemented using a computer game engine [22] [14] [8]. In this virtual fair, the 3D game engine provided by Adobe Director 11.5 has been used. This software offers the programmer many features as

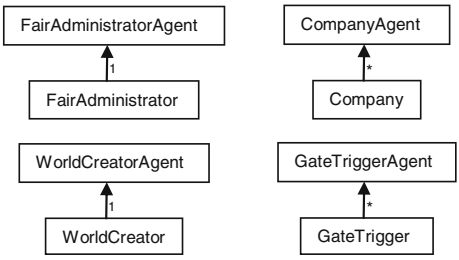

Fig. 15 Agent Model of the virtual trade fair

Table 2 Service model for the *FairAdministratorAgent* according to the *liveness* properties defined in Fig. 12

Service	Inputs Outputs	Pre-condition	Post-condition
Registration service	companyInfo companyPos registrationOK	true	true
Tour service	companyInfo tourOK	registrationOK=true	true
GuidedTour service	companyInfo listStands	registrationOK=true	tourOK=true ⇒ $_{\text{listStands} \neq \emptyset}$
AssignStand service	companyInfo companyPos standOK	registrationOK=true	true
PublishMultimedia service	companyInfo MMelements publishOK companyRecord	registrationOK=true ∧ standOK=true	true
ChatSession service	companyInfo	true	true

well as a powerful scripting language called LINGO, which enables interaction with external files and certain Windows APIs. This makes it possible for the virtual world to communicate with the web.

Some data from the exhibiting companies are accessible from the stand by clicking on some interactive buttons created on the main wall (Fig. 17). The stand's interaction has been implemented using LINGO language. In order to retrieve the information from the appropriate website address, PHP and MySQL have been employed to implement a query to the data base. The streamed pictures of the companies have been converted into a video and a slide show using Flash Action Script 2.0.

7 Case of study

This section presents a detailed use-case that illustrates the behavior of our agent-based solution. Our testing scenario consists of a virtual fair composed of 6 pavilions with 36 stands inside each one.

This use-case was run on an Intel Core i7 2.8 GHz CPU, 6 GB RAM running on Windows 7. The graphics hardware used was NVIDIA GeForce 480GTX. The fair has been developed as a web application with Adobe Director so that it can run on the major web browsers.

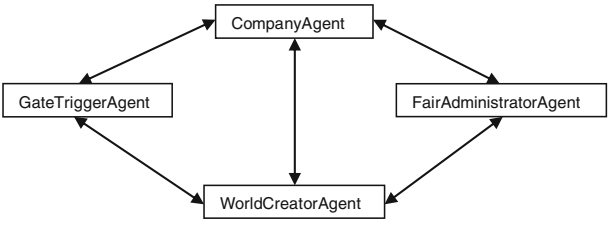

Fig. 16 Acquaintance model

Fig. 17 An image of a stand where the multimedia interactive buttons can be appreciated

Although we tested the application on Mozilla Firefox, Google Chrome and Internet Explorer, our tests show that the performance obtained was almost identical. The tests were run on a 3 GB/s bandwidth LAN network.

For our use-case, we simulate a usual situation that occurs in a trade fair. A user that represents a company accesses the virtual world and, after logging into the system, its avatar appears in its stand. After a few seconds, this user freely moves around the pavilion visiting different stands. In a particular stand, the user accesses the available multimedia material of the exhibiting company and spends 60 s consulting this material. Next, he/she walks towards the pavilion door to get to the outdoor scenario. Once in the main corridor, the user moves around for 3 s and then he/she asks for a guided tour in order to visit the stands of the companies that sell a product that he/she is interested in. When this user is allowed to perform the selected tour, the avatar automatically changes position and appears in the stand of the selected company, the first one in the tour.

The agents involved in these situations and the processes that are performed are detailed below.

1. The simulation begins when the *CompanyAgent CoA1* logs into the system and appears in the stand that was previously assigned to him/her in the registration process. This action is attended by the *FairAdministratorAgent FaA*, that checks if the company is allowed to access this virtual environment, and by the *WorldCreatorAgent WcA* that creates the appropriate view of the world.
2. Once *CoA1* is in the pavilion, this company's agent moves around and freely visits the different stands in this virtual building. The agent spends 15 s in this free visit.
3. Then, *CoA1* consults the multimedia material of the exhibiting company agent *CoA2* for 60 s (Fig. 17). This multimedia material is showed by the *MediaAgent MeA*, not explained in this article for reasons of brevity. It overlaps the current view of the virtual world that is obtained by the *WorldCreatorAgent WcA*.
4. Next, *CoA1* walks towards the pavilion door for 3 s to go out of the pavilion and get into the outdoor scenario. In this action, a *GateTriggerAgent* (*GtA*) is implicated. This is the agent in charge of asking the *WorldCreatorAgent WcA* for a new view of the virtual trade fair it the user demand is accepted.
5. Once in the outdoor scenario, the user moves around for 5 s. Again, the *CoA1* agent and the *WorldCreatorAgent WcA* are involved in this action.
6. Finally, *CoA1* asks for a guided tour by sending a request to the *FaA*, while a view of the outdoor environment is being rendered by the *WorldCreatorAgent WcA*.
7. Following this route, *CoA1* visits the stand of the exhibiting company *CoA3*, the first one in the list stand route. Then, CoA1 is directly moved to the requested stand. Again, the *WorldCreatorAgent WcA* is in charge of obtaining the current view of the world.

Table 3 summarizes all of the processes involved in this simulation. It analyzes every situation, providing information about the duration of every action, the agents involved, the

Table 3 Processes developed by the agents involved in the detailed actions

	Action	Duration of this action	Agents involved	Processes	Average Time (seconds)
1	CoA1 accesses to the virtual world	5 s	CoA1	RegistrationRequest	0.2
			FaA	RegistrationResponse	0.15
			WcA	CreateView	0.02
2	CoA1 freely moves around the pavilion	15 s	CoA1	MovetoRequest	0.18
			WcA	CreateView	0.02
3	CoA1 checks the multimedia material in the CoA2 stand	60 s	MeA	MMResponse	0.02
			WcA	CreateView	0.02
4	CoA1 gets out the pavilion toward the outdoor scenario.	10 s	CoA1	CrossDoorRequest	0.03
			GtA	ConmuteWorldRequest	0.25
			WcA	CreateView	0.02
5	CoA1 freely moves around the outdoor scenario.	5 s	CoA1	MovetoRequest	0.18
			WcA	CreateView	0.02
6	CoA1 asks for a guided tour	4 s	CoA1	GuidedTourRequest	0.18
			FaA	ListStandsResponse	0.03
			WcA	CreateView	0.02
7	CoA1 moves to a stand in the stand list route.	5 s	CoA1	MoveToRequest	0.18
			WcA	CreateView	0.02

processes that these agents execute in order to perform it and the average time of these processes.

As shown in the table, every situation is simulated by performing more than one process. Nevertheless, some of them are continuously being executed, such as the *CreateView* process, while others only are performed on demand, such as *CrossDoorRequest* or *GuidedTourRequest*. In the table, the average duration of every process has been informed.

In our test, the virtual trade fair is rendered in real time. It averages to 30 frames per second (fps) during the simulation. The execution of the processes that the different agents perform does not reduce the frame rate. This is because most of these processes are executed in background while the virtual scenario is visualized.

Thus, we can conclude that the interactive frame rate is not reduced although the system has to attend to the different requests of the agents. The execution of these processes does not slow the real time visualization of the system.

8 Conclusions and future work

In this article, we have described a virtual trade fair following an agent-based methodology for the analysis and design of interacting systems. The virtual world has been implemented using some graphics accelerations techniques that allow the virtual scenarios to be rendered in real time. In 3D environments, particularly in adaptive virtual ones, the agent is a very important element, since it also stores the behavior of the user; his/her style of navigation; forms of interaction with the environment; and the evolution of the user's learning within the system. Thus, agents are able to carry out the decision-making process for generating virtual worlds and eventual adaptations using the information contained in the agent-based model [1] [17].

In this paper, these concepts have been analyzed and applied to an interactive business-to-business environment [2] [6]. This virtual trade fair has been specified and designed and every interaction between the identified agents and with the environment has been detailed. Gaia has been the chosen methodology in the design of this application [27]. Moreover, this economic virtual world has also been implemented, following the presented analysis. This agent-oriented approach has made it easy for developers to create a virtual trade fair, because it has mainly been based on the interaction of the identified agents in this virtual environment.

Our current research work is focused on proposing more specific guidelines and conceptual tools to support engineers with some others implementations of the presented analysis. Moreover, we are trying to develop a study related with how the changes in the design are reflected in the implementation and what different problems may arise at this level and that we have still not identified.

Acknowledgments This work was supported by the Spanish Ministry of Science and Technology (Project TIN2010-21089-C03-03) and Feder Funds.

References

1. Argente E, Botti V, Carrascosa C, Giret A, Julian V, Rebollo M (2011) An abstract architecture for virtual organizations: The THOMAS approach. Knowl Inf Syst 29(2):379–403
2. Berger, H., Dittenbach, M., Merkl, D., Bogdanovych, A., Simoff, S., Sierra, C. (2006) Playing the e-business game in 3d virtual worlds. In: OZCHI'06: Proceedings of the 18th Australia conference on Computer-Human Interaction, pp. 333–336. ACM
3. Businesswomanfair (2013) http://www.businesswomanfair.com/ Accessed March 2013
4. Cernuzzi, L., Juan, T., Sterling, L., Zambonelli, F. (2004) The gaia methodology: Basic concepts and extensions. In: Methodologies and Software Engineering for Agent Systems. Kluwer Academic Press
5. Companies turn to virtual trade shows to save money (2013) http://usatoday30.usatoday.com/travel/news Accessed March 2013
6. Czerniawska F, Potter G (1998) Business in a Virtual World: Exploiting Information for Competitive Advantage. Wiley, Basingstoke
7. Duke, R., Rose, G.: Formal Object Oriented Specification Using Object-Z. (2000). Cornerstones of Computing Series. Palgrave Macmillan Limited
8. Eberly, D. (2000) 3D game engine design: a practical approach to real-time computer graphics. Morgan Kaufmann Publishers Inc
9. Garcés A, Quirós R, Chover M, Camahort E (2010) Implementing virtual agents: Haba-based approach. Int J Multimed Appl 2:1–15
10. Garcés, A., Quirós, R., Chover, M., Huerta, J., Camahort, E. (2007) A development methodology for moderately open multi-agent systems. In: Proceedings of the 25th conference on IASTED International Multi-Conference: Software Engineering, SE'07, pp. 37–42
11. Globbarea (2013) http://www.globbarea.com/en/Home Accessed March 2013
12. Itradefair.com (2013) http://www.itradefair.com Accessed March 2013
13. Marketplace365 (2013) http://www.marketplace365.com/Accessed March 2013
14. Noh, S.S., Hong, S.D., Park, J.W. (2006) Using a game engine technique to produce 3d entertainment contents. In: Proceedings of the 16th International Conference on Artificial Reality and Telexistence–Workshops, ICAT'06, pp. 246–251
15. On24. (2013) http://www.on24.com/products/virtual-environments/ Accessed March 2013
16. Prendinger H, Ullrich S, Nakasone A, Ishizuka M (2011) Mpml3d: Scripting agents for the 3d internet. IEEE Trans Vis Comput Graph 17(5):655–668
17. Ranathunga, S., Cranefield, S., Purvis, M. (2011) Interfacing a cognitive agent platform with a virtual world: a case study using second life (extended abstract). In: Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems, pp. 1181–1182. International Foundation for Autonomous Agents and Multiagent Systems
18. Remolar I, Chover M, Quirós R, Gumbau J, Castelló P, Rebollo C, Ramos F (2011) Design of a multiuser virtual trade fair using a game engine. Trans Comput Sci 12:118–139
19. Second life (2013) http://secondlife.com/whatis Accessed March 2013

20. Shoham Y (1993) Agent-oriented programming. Artif Intell 60(1):51–92
21. Smitht G (2000) The Object-Z specification language. Kluwer Academic Publishers, Norwell
22. Trenholme D, Smith S (2008) Computer game engines for developing first-person virtual environments. Virtual Reality 12(3):181–187
23. Unisfair (2013) http://www.unisfair.com/ Accessed March 2013
24. Vijaykar, S., Kadavasal, M.S., Dhara, K.K., Wu, X., Krishnaswamy, V. (2009) Virtual worlds as a tool for enterprise services. In: CCNC'09: Proceedings of the 6th IEEE Conference on Consumer Communications and Networking Conference, pp. 186–190. IEEE Press
25. Vosinakis S, Panayiotopoulos T (2005) A Tool for Constructing 3D Environments with Virtual Agents. Multimed Tools Appl 25(2):253–279
26. Wooldridge, M., Ciancarini, P. (2001) Agent-oriented software engineering: The state of the art. pp. 1–28. Springer-Verlag
27. Wooldridge M, Jennings NR, Kinny D (2000) The gaia methodology for agent-oriented analysis and design. Auton Agent Multi-Agent Syst 3(3):285–312
28. Xpofairs (2013) https://www.xpofairs.com/ Accessed March 2013
29. Zambonelli F, Jennings NR, Wooldridge M (2003) Developing multiagent systems: The gaia methodology. ACM Transaction Software Eng. Methodol 12(3):317–370

Dr. Inmaculada Remolar received her PhD in Computer Science from the Universitat Jaume I, in Castellón (Spain) in 2006. Currently she is professor at the Department of Computer Languages and Systems at the same university. Her research areas include multiresolution modelling, real-time visualisation of trees and real-time shadow rendering.

Alejandro Garcés received his PhD degree in Computer Science at the Universitat Jaume I, Spain, where he is Research Assistant at Institute of New Imaging Technologies. His current research interest is focused on Software Engineering for the development of Multi-Agent Systems and virtual worlds

Cristina Rebollo is a professor at the Universitat Jaume I of Castellón in Spain. She received her MS degree in Computer Science in 1988 from the Universitad of Deusto of Bilbao, Spain. She received her PhD in Computer Science from the Universitat Jaume I of Castellón, Spain in 2006. She is currently working at the Department of Computer Languages and Systems at the University Jaume I. Her research areas include multiresolution modelling, real-time visualisation of trees and real-time shadow rendering.

Miguel Chover is full professor in the Department of Computer Languages and Systems at the University Jaume I of Castellón. He received a Ph.D. in computer science from the Polytechnic University of Valencia in 1996. He is currently director of the Degree in Video Game Design and Development, director of the Center for Interactive Visualization and Secretary of the Institute of New Imaging Technologies at the University Jaume I. His research interests include geometric modeling, interactive visualization and game technology. He is president and active member of the Spanish Association for Computer Graphics that belongs to EUROGRAPHICS.

Dr. Ricardo Quirós received his PhD degree in Computer Science at the Technical University of Valencia in 1996. Currently he is co-director of the Centre for Interactive Visualization, and professor of Information Systems at the Universitat Jaume I. His current research interest is focused on Computer Graphics and Multimedia, especially in Virtual and Augmented Reality, Light Field Rendering, Auto Stereoscopic Visualization and 3D Television. He is member of the European Association for Computer Graphics (Eurographics) and reviewer for local and European conferences and journals (Spanish Conference on Computer Graphics, Ibero-American Symposium on Computer Graphics, Computers & Graphics Journal).

Jesus Gumbau received his MS degree in Computer Science in 2004 from the Universitat Jaume I of Castellón, Spain. He received his PhD in Computer Science from the same university in 2011. He is currently working at the Department of Computer Languages and Systems at the University Jaume I. His research areas include multiresolution modelling, real-time visualisation of trees and real-time shadow rendering.