IMPLEMENTING VIRTUAL AGENTS: A HABA-BASED APPROACH

Alejandro Garcés[1], Ricardo Quirós[2], Miguel Chover[3] and Emilio Camahort[4]

[1, 2, 3] Department of Computer Languages and Systems, Jaume I University, Castellón, SPAIN
[1]agarces@uji.es
[2]quiros@uji.es
[3]chover@uji.es

[4]Department of Computer Systems, Polytechnic University of Valencia, Valencia, SPAIN
[4]camahort@dsic.upv.es

ABSTRACT

In recent years developments in gaming technology have focused on Massively Multi-User Persistent Worlds. This virtual communities host thousands of users interacting with each other through the Internet in real time. Such communities require new distributed programming paradigms for implementing the virtual life management. In this work we present a practical methodology to add Virtual Agents to Massively Multi-User Persistent Worlds. To demonstrate the use of our methodology we present a real-world application. It adds a set of virtual agents to a 3D chat. The agents survey the users to obtain data to evaluate the processes and components of the virtual world.

KEYWORDS

Software Agents, Multi-agent systems, Agent-Oriented Software Engineering, Virtual Environments

1. INTRODUCTION

The advances in the new century are causing that complex virtual spaces becomes a reality. There exist a vast number of Virtual Environments (VEs), each of them bringing its own vision, with a set of tools [1]. We would define a Virtual Environment (VE) as a three-dimensional computer representation of a space in which users can move their viewpoint freely in real time [2].

In the last years, developers have focused on Massively Multi-User Persistent Worlds (MMPWs). This virtual communities host thousands of users interacting with each other through the Internet in “real” time. MMPW fall into two broad categories: games like Everquest, Ultima Online, and Asheron’s Call, and virtual environments or metaverses like Second Life, Active Worlds or Blaxxun [3]

This new kind of entertainment software is causing new and important advances in certain technologies; for example the Computer Graphics, the Simulation, the Artificial intelligence, the Distributed Systems and the Graphical User Interfaces. These innovations have not been only into the field of the computer science. These advances are going beyond the world of the entertainment and they are causing a revolution in the development of VEs in the network with many applications for the education, the commerce, the simulation, among others.

Users are not the only living objects inside an MMPW. Non-Player Characters (NPCs) also co-exist with regular users, and need their own social behavior. This behavior must be described at a semantic level, a level that allows control of the environment's complexity and dynamic evolution. Intelligence implementations can be programmed using autonomous components (agents) that are simpler but use more complex communication protocols.

Agents and multi-agent systems, other than a technology, represent a brand new paradigm for software development [4]. This paradigm adopts agents as the basic conceptual components of software systems.

Many researchers are focusing on the human models for the development of Multi-Agent Systems (MASs). They introduce very abstract theories, which can be successfully used for the development of some open and complex environments [5]. However, these models are often inefficient for others real-world applications; for example the web computing.

These issues limit the development of simple practical prototypes of MASs. To address this problem we define a restrictive class of MASs. It is named *Moderately Open Multi-Agent Systems* (MOMASs). It includes a Gaia-based process for the development of MOMAS systems called *Homogeneous Agents Based Architecture* (HABA). Our model and development process have the advantage that they allows both high- and low-level behavior specifications.

To demonstrate the use of our methodology we present a real-world application. It adds a set of virtual agents to a 3D chat. The agents survey the users to obtain data to evaluate the processes and components of the virtual world. The whole development process is described by means of stepwise refinement from the analysis stages to the system implementation.

The remainder of our paper is organized as follows. In the background section we present related work focusing on Gaia methodology for the analysis and design of MASs. In Sections 3 we introduce our model and development process. Section 4 describes our real-world application. We conclude our paper with conclusions and directions for future work.

2. BACKGROUND

Many different architectures and methodologies for the development of agent-oriented systems have been presented in the specialized literature: for example BDI [6], MAS-CommonKADS [7], MaSE [8] and Ingenias [9], among others. However, in this work we are particularly interested in Gaia methodology [10]. A review of AOSE methodologies is beyond the scope of this paper. The reader can see some important papers, which survey many of these methodologies (for example, [11] and [12]). Also, Cernuzzi, Cossentino and Zambonelli [13] focus on process models for the agent-oriented software development. They survey the characteristics of a number of agent-oriented methodologies, as they pertain to software processes. Winikoff [14] briefly review the current state of play in the area of agent-based software engineering, and then he consider "what next?".

The MAS technology has been applied by some practitioners to implement intelligence in games or virtual environments. Barella et al. [15] include intelligence in computer games using the JADE platform. Davies et al. [16] use BDI to specify the behavior of Virtual Agents. Also, there are many works about agents and multi-player online games (for example, [17], [18], [19], [20]). A very important approach is related with the Agent Organizations [21] [22]. Some methodologies for development of MASs allow this approach. One of them is the Gaia methodology for the analysis and design of MASs.

Gaia was the first complete methodology for the analysis and design of MAS. It is based on the organizational metaphor. The system is conceived like a computational organization of agents. Each agent has certain roles in the organization. The agents cooperate to achieve the common objectives of the application. Gaia deals with both the macro (societal) level and the micro (agent) level aspects of design.

Instead of an implementation framework Gaia only defines a conceptual framework suitable for analyzing and designing agent-based systems. Gaia has certain disadvantages from a practical point of view. Its design methodology is so abstract that it does not provide a clear relationship between the design specification and the final system implementation.

There are a lot of theoretical studies of Gaia [5] [23], all emphasizing its strengths and weakness. Moreover, Akbari and Faraahi [24] present an evaluation framework for agent-oriented methodologies. They have used this framework to evaluate the Gaia methodology to identify its principal characteristics. Its limitations have fostered the development of many variants of Gaia. For example, in [23] three different variants are presented. They overcome some conceptual problems and extend the developing of MAS to complex open environments. Still, none of them proposes how to implement MAS on real platforms. Independently of these weaknesses, GAIA is considered one of the most promising methodologies for agent-oriented analysis and design. In the context of our work, GAIA focuses on the use of high-level abstractions, which are based on the organizational metaphor. It is very important for the development of a MMPW.

3. MODERATELY OPEN MULTI-AGENT SYSTEMS

The MOMAS technology has been developed for a class restrictive of Multi-Agent Systems. MOMAS model focuses on a class of software agents: the stationary agents. These software agents execute only on the system on which it begins execution; they may typically use a communication system such as remote procedure calling. A stationary agent is implemented as a code component and a state component. Agents need an execution environment at all involved computers.

3.1. Multi-Agent System Architecture

Our programming environment allows prototyping MOMAS with the below architecture (figure 1). Applications have a static structure since agent classes and their relationships do not change during execution. Services provided by the agents are also static. Agents are clustered into communities called packages. A management module within the MAS handles the life cycle of packages and agents, their communication, and the social state's public information. This module is a special agent that interfaces between the social state, the agents and the user in charge of running the system [25].

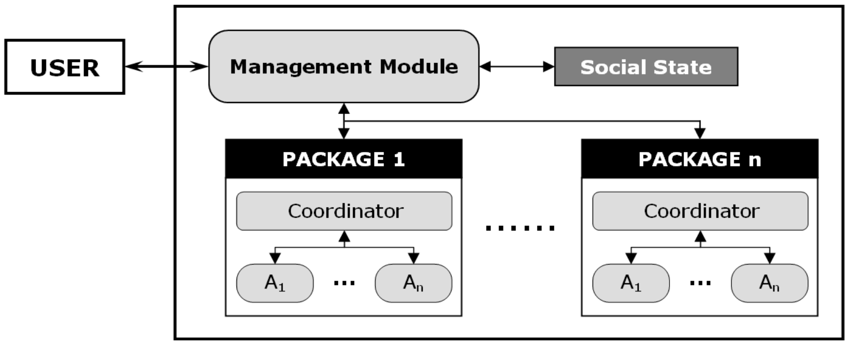


Figure.1. Architecture of a Moderately Open Multi-Agent System

3.2. The HABA Development Process

To overcome the problems of Gaia and its extensions we propose a new MAS development process called HABA [26]. This process can be used for the analysis, design and implementation of agent-oriented systems. Its main difference from GAIA is the type of systems they can model. HABA can model MOMAS as defined in previous section. The three main concepts in HABA are: Roles, Agent Classes and Agents. Roles are the basic modeling structure. Agent Classes are made of roles, and Agents are instances of Agent Classes. The HABA.DM methodology for the analysis and design of systems, the HABA.PL programming language, and the HABA.PM project manager are the structural elements of the HABA development process we propose. The relation between the different HABA models is shown in figure 2.

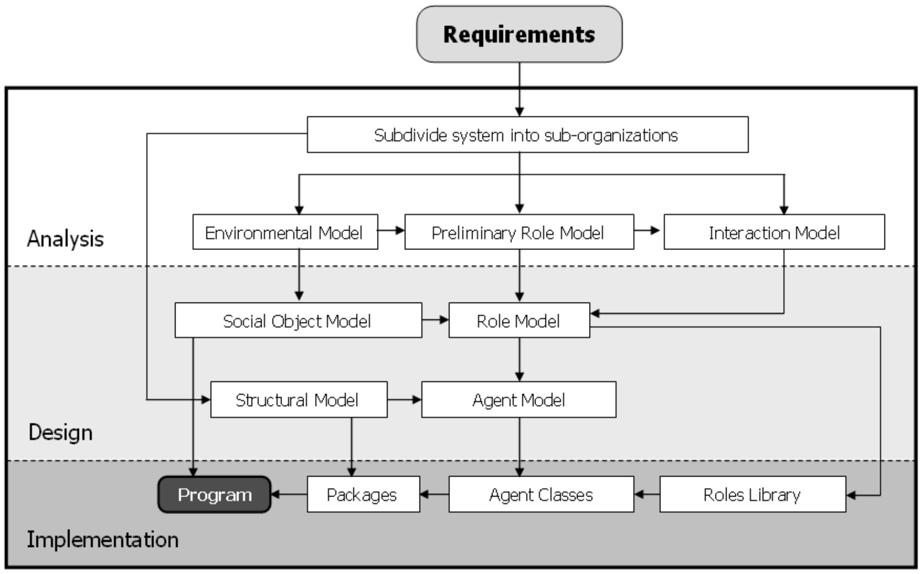


Figure 2. Models in the HABA Process

3.2.1. Analysis and Design of MOMAS: The HABA.DM Methodology

ENVIRONMENT: *<MOMAS name>*
OBJECTS: *<Environmental Objects>*
CONSTRAINTS: *<Constrains on Objects>*

(a)

ROLE SCHEMA: *<Role name>* **in** *<Subdivision>*
DESCRIPTION: *<Short description of role>* **PROTOCOLS & ACTIVITIES***:* *<Protocols and activities in which the role plays a part>* **PERMISSIONS:** *<Rights associated to the role>* **RESPONSIBILITIES:** **Liveness** *<Liveness responsibilities>* **Safety** *<Safety responsibilities>*

(b)

Figure 3. (a) Components of Environmental model. (b) Components of Preliminary role model.

In the analysis stage we obtain a set of models that represent the organization structure, the environment, a preliminary description of the roles, and the interaction protocols of the MOMAS [27]. Modeling the environment implies identifying its basic features, the resources that can be found in the environment, and the way via which agents can be interact with it. The environmental model (see figure 3a) contains the computational resources of the MAS.

Our preliminary role model (see figure 3b) defines the generic behavior of entities within an organization, and it provides a high degree of reusability. The interaction model (see figure 4) is like Gaia's model, but has some subtle differences from the semantic point of view. Due to the scope of our methodology, we limit the interaction between components to controlled ask-reply communications. This is the kind of interaction supported by the programming language HABA.PL.

<Protocol Name>		
<Sender>	*<Receiver>*	*<input>*
<Description>		*<output>*

Figure 4. Components of Interaction model.

The analysis stages can be summarized by means of the following steps:

(i) Identify the goals of the organization goals and its subdivision in sub organizations.

(ii) Identify the environmental model of the MOMAS, defining computational objects (variables) and constraints between them.

(iii) Build the preliminary role model using the environmental model,. This process is similar to the one performed in GAIA, but we associate a global or local scope to each role according to the subdivision of the organization.

(iv) Identify protocols in the preliminary role model and build the interaction model. Each protocol defines an interaction between two roles. More complex interactions between roles and the environment must be modelled with the shared resources of the MOMAS environment (Social State, see figure 1).

In HABA.DM, the design stage produces the social objects model, the role model, the agent model and the structural model [27]. A social object model (see figure 5a) describes the information resources that belong to the social state of the MOMAS. These resources are types, variables and constraints managed by the MOMAS Management Module (see figure 1). The social object model is derived from the environmental model produced by the analysis stage.

SOCIAL OBJECTS: *<Momas Name>*
TYPES: *<Types on design>*
OBJECTS: *<Environmental Objects>*
CONSTRAINTS: *<Constrains on Objects>*

(a)

ROLE: *<Role Name>* **in** *<Subdivision>*	
DESCRIPTION: *<Short description of role>*	
SERVICES: *<Service schemata>*	
ACTIVITIES: *<Activity schemata>*	
PERMISSIONS: *<Rights associated to the role>*	
RESPONSIBILITIES	
Liveness	*<Liveness responsibilities>*
Safety	*<Safety responsibilities>*

(b)

Figure5. (a) Components of Social objects model. (b) Components of Role model

The role model (figure 5b) describes each role of the MOMAS using four basic components: the activities, the services, the properties and the permissions. Properties can be active or safe. This role model derives directly from the preliminary role model and the interaction model.

The agent model (figure 6a) specifies the agent classes used in the MOMAS. Instances of these classes are the agents that the system creates at runtime. Figure 6b shows how the MOMAS system is built using packages.

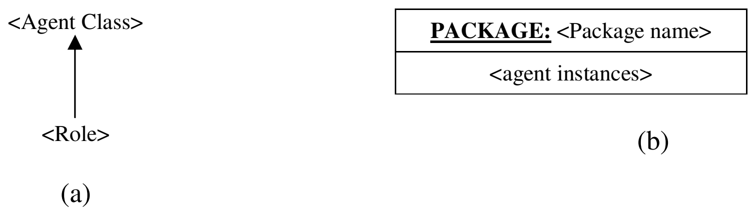


Figure 6. (a) Agent model. (b) Structural model: A Package

The design stages can be summarized by means of the following steps:

(i) Given environmental model produced by the analysis stage, create the social object model.

(ii) Given the preliminary role model and the interaction model build the role model defining the different services and their potential clients.

(iii) Create the agent model as by creating the agent classes, and assigning roles to the different agent classes.

(iv) Build the structural model, grouping the agents in packages according to the MOMAS architecture.

3.2.2. Development framework

The HABA process presents a framework for fast prototyping of MOMAS systems. It provides a language to program roles, agents, social state and communication protocols, which is named HABA.PL. It allows the definition of all components and the centralized execution of the systems (for more detail [25]).

4. MODELLING AN INTELLIGENT VIRTUAL ENVIRONMENT

Adding AI to Virtual Agents (VAs), is the most important application of AI to MMPW. The VAs can be introduced into a layer in the design of a whole MMPW [28].

We describe and demonstrate the use of MOMAS model and development framework adding a set of agents to a 3D Chat. Our goal is to survey the users of the chat about the quality of the services of an educational institution. The results of the surveys are used to plan organizational improvements. We design a set of questions that the users can freely answer in different ways.

In order to incorporate intelligence in the virtual world we use our MOMAS development model and framework. We included an abstract layer (see Figure 7), which is an MOMAS system. This layer has a static structure since agent classes and their relationships do not change during execution. Services provided by the agents are also static. Agents are clustered into communities called *packages*. A *management module* within the MAS handles the life cycle of packages and agents, their communication, and the social state's public information. This module is a special agent that interfaces between the social state and the agents in charge of running the MOMAS subsystem. Moreover, it allows the interaction among the virtual agents and the users through of the chat server application.

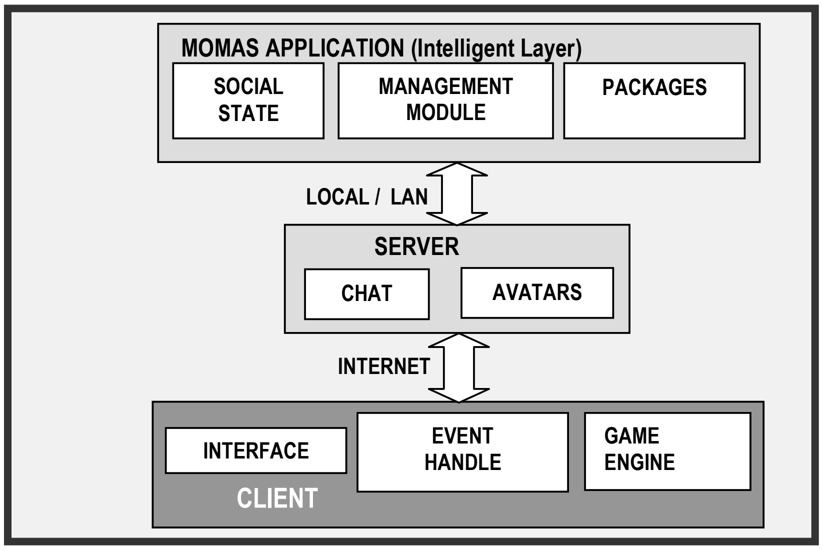


Figure 7. Architecture of an intelligent MPMM

The 3D chat has a client-server architecture [29]. The user runs the client application that is in charge of rendering and communication with two server applications. One of those servers manages the positions of all the avatars. It is called Avatar Location Server. The other server, called Chat Server, handles the Chat services. Virtual agents are organized in the MOMAS application that also communicates with these two servers. Communication between the MOMAS application and the servers allows the following features. Using avatars it allows customizing the abstract life of the agents inside the environment. It also allows feedback from the environment to the MOMAS application.

The Chat server uses a telnet connection. The Avatar Location Server monitors the state of each client (name, position, orientation) and sends that information to the other clients of the servers: human users (players) or virtual agents (non-players from the MOMAS application). The client application is in charge of rendering the world by drawing the avatars in their positions. It also handles the communication with the servers to show the events happening in the Chat.

4.1. Application Description

Here we describe and demonstrate how our application works combining a set of agents from our MOMAS system with a 3D chat. The goal is to survey the users of the chat about the quality of the services of an educational institution. The results of the survey will be used to develop an improvement plan for the institution’s organization. We have a set of questions that the users can freely answer using either natural language or choosing from multiple choices. The questions are asked by a set of individuals (surveyors) that are represented by avatars in the virtual world. The actions of these surveyors correspond to autonomous agents with a high level of abstraction running inside the MOMAS application layer.

Collecting Data from the Application Domain

The virtual world is a university campus where the users of the 3D chat move around. Its size is big and its topology complex. It contains multiple teaching areas, as well as academic service areas like, for example, a library. In some of these areas, like a classroom, surveying is not appropriate. Therefore, our example only considers leisure areas where the users can be approached in an informal way (see Figure 8).

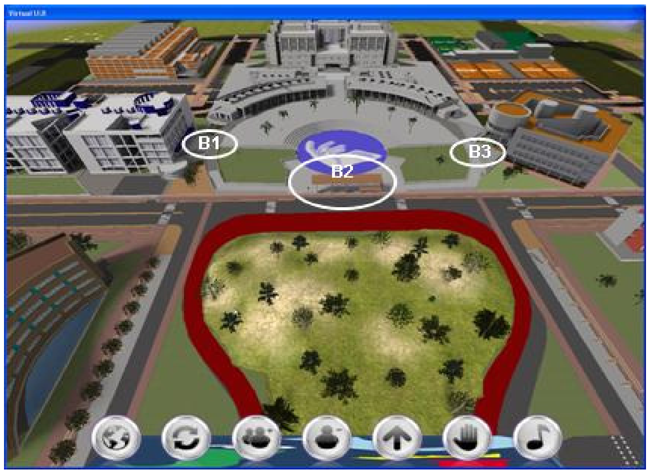


Figure.8. Partial view of the virtual campus

Let B=<B_1, B_2.....B_n> be a the set of regions of interest where B_i=(C_i, ρ_i) (i= 1..n), C_i is the center of region B_i and ρ_i is the radius of the vicinity of B_i.

Let P={p | p $\in$ NICK $\times$ $\Re^3$} be a set of pairs nickname-position, where NICK (NICK $\subseteq$ seqCHAR) is the set of all possible nicknames of a user in the virtual world, seqCHAR is the set of all possible character strings and $\Re$ is the set of real numbers. P contains elements with the form <nickID, <x, y, z>>.

Let S = <S_1, S_2,…, S_m> be the array that contains the m surveys where S_j=<sID,*l*,Q,R> (j =1..m) is a 4-tuple that stores the information of the j-th survey, identified by sID, a natural number, *l* is the number of questions in the survey, Q=<Q_1,Q_2,.....Q_l> is an *l*-tuple of questions and R= < R_1,R_2,....R_l> is the *l*-tuple that contains all the classified information about the replies of the users to survey sID.

Specifically, for all k such that k = 1..*l*, R_k={r | r $\in$ seqCHAR $\times$ $\aleph$ $\times$ $\aleph$} is the set of all answers to question Q_k of survey nID, where the elements r of R_k have the structure r = <replySTRING, regionID, userID>, where ReplySTRING is the text of the reply given in region regionID by user userID.

To simplify decision making during the survey, we need to compute the effectiveness of the surveys in the different regions. To compute effectiveness we take the ratio between the answers to a question and the number of times the question was asked. So we have a matrix E=[$E_{i,j,k}$] (i=1..n; j=1..m; k=1..*l*), where element $E_{i,j,k}$ contains the effectiveness of question Q_k of survey S_j inside region B_i.

$$E(i,j,k) = \frac{\#\rho(i,j,k)}{\eta(i,j,k)}$$

Now, for all i, j, k, if S_j = <sID, *l*, Q, R> and π_k(R) is the k-th component of tuple R, then

$\rho(i,j,k)$ = {reply | exists uID $\in$ $\aleph$, such that <reply, i, uID> $\in$ π_k(R)};

$\#\rho(i,j,k)$ is the cardinal of the set $\rho(i,j,k)$; and

$\eta(i,j,k)$ is the number of times the k-th question of the j-th survey was asked into i-th region.

How the Application Runs

In the application we distinguish three different role types: the surveyor, the observer and the analyst. The flow chart of the survey is shown in Figure 9. The observer starts the process, running the surveyors (M_0). Then, the observer selects areas adequate for surveying and informs the surveyors (M_1). The surveyors, given a target area and their experience, compute a move and determine the position they will move to in order to ask users. This new position is notified to the observer for control reasons (M_2). Once the surveyor completes the number of allocated surveys, it informs the observer (M_3). Once all the surveyors have finished, the observer instructs the analyst to start processing the answers to the survey (M_4).

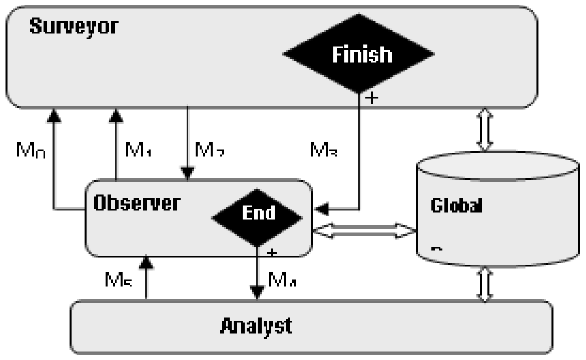


Figure 9. Message flow chart of the survey process

In the following sections we describe the analysis and design of the survey system using the HABA.DM methodology. We describe the role-driven process by stepwise refinement of the surveyor role. The other roles can be modelled in the same way.

4.2. Analysis and Design of MOMAS Layer

The HABA.DM methodology for the analysis and design of MOMAS allows fast modelling of agent-based systems in an incremental process based on roles. It provides models to capture the complexity of the information of the environment and the system architecture.

Analysis Stage

The Environment Model (figure 10a) contains the objects and the social constraints of the survey system. Constraints are expressed in Z notation. The environment contains the partitioning of the world into regions, the positions of the users and virtual agents, the questions of the survey, and the replies to the survey. Additionally, information is recorded associated to each region in order to determine the per-region effectiveness of the survey. The effectiveness information is used to handle the movements of the surveyors.

The survey system defines three roles: observer, surveyor and analyst. Figure 10(b) shows the preliminary role model of the surveyor role. The goal of this role is survey the users of the 3D chat. It waits for a move suggestion by the observer. Then, it evaluates it using its own experience. Finally, the surveyor role reports its decision to the observer and moves to an optimal position to interview users. During the surveys, the surveyor randomly selects questions and collects the answers of the users. The role modifies the Response object of the environment and access the Question, Region and Effectiveness objects. Its safety property establishes the maximum time a surveyor waits in a region to obtain its responses. The remaining roles of the system can be modelled in the same way.

ENVIRONMENT : SurveySystem

OBJECTS

Survey ≅ [id_Survey]
Question ≅ [id_Question ; id_Survey]
Region ≅ [id_Region]
Answer ≅ [id_ Question]
Position ≅ [id_Avatar]
Efectiveness ≅ [id_Region]
(*…. Other Objects ……*)

CONSTRAINTS

$\forall$ a,b: Survey | $a \neq b \bullet$ a.id_Survey $\neq$ b.id_Survey
$\forall$ a,b: Region | $a \neq b \bullet$ a.id_Region $\neq$ b.id_Region
$\forall$ a,b: Question | $a \neq b \bullet$ a.id_Question $\neq$ b.id_Question
$\forall$ a,b: Position | $a \neq b \bullet$ a. id_Avatar $\neq$ b. id_Avatar
$\forall$ a,b: Efectiveness | $a \neq b \bullet$ a. id_Region $\neq$ b. id_Region
(*…. Other Constraints …..*)

(a)

ROLE SCHEMA: Surveyor

DESCRIPTION:

Survey the users.

PROTOCOLS & ACTIVITIES:

AwaitSuggestMovement, EvaluateMovement,
InformEvalMovement, Move,
SelectQuestion, SendQuestion,
AwaitAnswer, WaitRegión,
Finish, InformFinish

PERMISSIONS

changes Answer
reads Question, Region, Efectiveness

RESPONSIBILITIES

Liveness
Surveyor = (MakeSurvey |
Finish. InformFinish)*
MakeSurvey = (AwaitSuggestMovement.
EvaluateMovement.
InformEvalMovement. Move.
(SelectQuestion. SendQuestion)$^+$
(AwaitAnswer | WaitRegión))*

Seguras
$(t_a - t_0) < t_e$

(b)

Figure 10. (a) Environment model: Survey system. (b) Preliminary role model: surveyor

Figure 11 shows the protocol *SuggestMovement*, started by the Observer role and replied to by the Surveyor role. Using this protocol the *Observer* informs the *Surveyor* about a suggested move to a given region.

SuggestMovement		
Observer	Surveyor	r : Request
Send the Move Suggestion		reply : Boolean

Figure 11. Interaction model: *SuggestMovement* protocol

The Design Stage

In our 3D chat, the social object model (figure 12a) is an abstract representation of the MMPW scene ($\aleph$ denotes the set of natural numbers, $\Re$ is the set of real numbers).

Our surveying system has three basic roles: the Observer role, the Surveyor role and the Analyst role. The observer views the whole scene and informs the rest of entities. The surveyors interact with the users to get answers from them. The analyst extracts semantic information from the surveys. The Surveyor role (figure 12b) is derived from the preliminary role model of the surveyor preliminary role (figure 10b) and interaction model (figure 11). The protocols are transformed in services or messages depending on the agent that starts the communication.

The agent model specifies the agent classes used in the MOMAS. Class instances are the agents that the system creates at execution time. This model is similar to the model defined in GAIA. Figure 13 shows an example definition of two types of agents: *ObserverAgent* and *SurveyorAgent*, with roles *Observer* and *Surveyor*, respectively. The * symbol implies that 0 or more instances of the *SurveyorAgent* class may be created. The 1 symbol implies that exactly one instance of the *ObserverAgent* class may be created.

SOCIAL OBJECTS : SurveySystem

TYPES

CHAR, POS, TIME
SURVEY_STATUS ::= processing | delivered | stopped | cancelled
(*… Other types …*)

OBJECTS

Survey ≅ [id_Survey : $\aleph$; sName : seqCHAR; status: SURVEY_STATUS]
Question ≅ [id_Question : $\aleph$; id_Survey : $\aleph$; text: seqCHAR]
Region ≅ [id_Region : $\aleph$; center: POS; radius: $\Re$]
Answer ≅ [id_Question: $\aleph$; t: TIME; id_Region: $\aleph$1; id_Avatar: $\aleph$1; reply: seqCHAR]
Position ≅ [id_Avatar : $\aleph$; p: POS]
Efectiveness ≅ [id_Region : $\aleph$; id_Question; effectiveness: $\Re^{[0,1]}$]
(*…. Other Objects ……*)

CONSTRAINTS

$\forall$ a,b: Region | a≠b • a.id_Region ≠ b.id_Region
$\forall$ a,b: Question | a≠b • a.id_Question ≠ b.id_Question
$\forall$ a,b: Position | a≠b • a. id_Avatar ≠ b. id_Avatar
$\forall$ a,b: Efectiveness | a≠b • a. id_Region ≠ b. id_Region
(*…. Other Constraints …..*)

(a)

ROLE: Surveyor

DESCRIPTION Survey the users.

SERVICES

SuggestMovement [Observer] : POS X $\aleph$ → Boolean

ACTIVITIES

EvaluateMovement : POS X $\aleph$ → POS
Move : POS → Boolean
SelectQuestion : → Question
SendQuestion : Question → Question
AwaitAnswer( ) : → Boolean
WaitRegión : → Boolean

PERMISSIONS

changes Answer
reads Question, Region, Efectiveness

RESPONSIBILITIES

Liveness Surveyor = (SuggestMovement(in p: POS; in n: $\aleph$).
EvaluateMovement(in p: POS; in n: $\aleph$; out p1: POS).
Observer:: InformEvalMovement (p1). Move(in p1: POS).
(SelectQuestion(out q: Question). SendQuestion(in q: Question))$^{+}$.
(AwaitAnswer( ) | WaitRegión( ))*
Safety $(t_a - t_0) < t_e$

(b)

Figure 12. (a) Social object model of the 3D Chat. (b) Surveyor role model

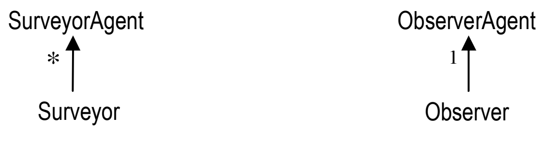
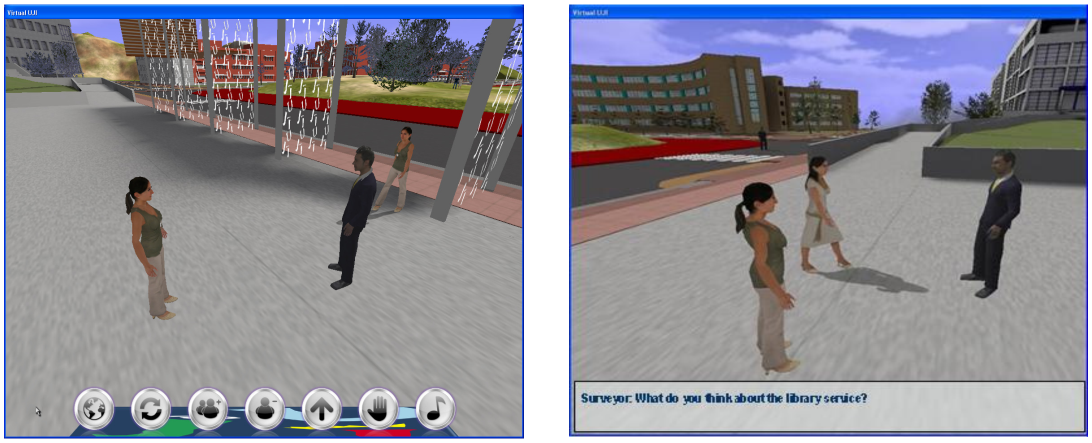


Figure 13. Agent model

4.3. Implementation and Results

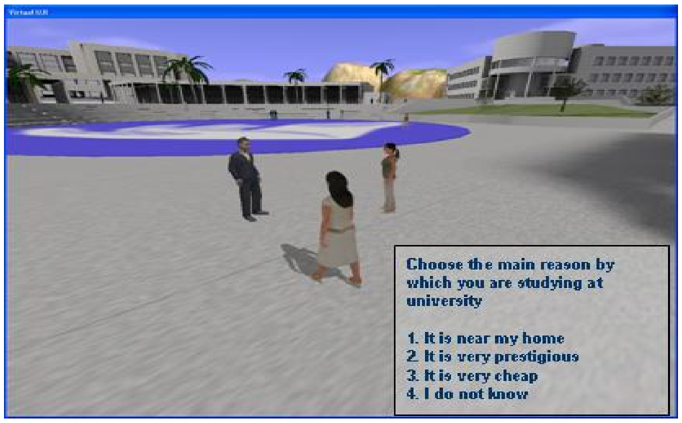
We have implemented the surveying system in a 3D Chat that simulates the campus of a real university. The application allows the user to walk through the world and to communicate with the different avatars using a chat service. The system has client-server architecture. The user executes a client application in charge of the visualization and the communication with two server applications. One of these servers holds the positions of all the avatars (Avatar Location Server), and the other manages the chat service (Chat Server).


Surveyor: What do you think about the library service?

(a) (b)

Figure 14. (a) a surveyor approaches a group of users, (b) performing a survey

The surveying module has been modeled with our MOMAS methodology, and has been implemented using our own programming language and project manager. The surveyors may interact with the different users in multiple ways. One of them uses a personalized avatar that moves around regions of the world, approaching users to ask them questions that can be answered using natural language. Figure 14 show two screenshots of our final application. In Figure 14(a) a surveyor moves towards a group of users to start a survey. In Figure 14(b) we can see how the survey is performed, with the surveyor asking the users their opinion about the library.


Choose the main reason by which you are studying at university

1. It is near my home
2. It is very prestigious
3. It is very cheap
4. I do not know

Figure 15. Dialog window: A surveyor interacts with an user

Another way a surveyor can interact with a user is by means of a dialog window. A dialog window allows more restricted questions, like multiple choice questions (see Figure 15). The control of a dialog window is made of three parts: (i) a portrait of the Surveyor, (ii) the actual survey question across the top of the dialog window, and (iii) the list of possible answers just below of the survey question. The user can interact in either of two ways. He or she can move away from the surveyor without answering its question. Alternatively, the user can reply to the question either typing some text or choosing an option of the dialog window.

5. CONCLUSIONS

Despite their advantages, agent-based models have not been widely used in the development of MMPWs. The reason is that they are too abstract and lack the tools needed to implement the system from the abstract specification. We have defined a development process for fast prototyping of a restrictive class of MAS. It is called Moderately Open Multi-Agent Systems. The main advantage of the proposal is that it allows both high- and low-level specification of Multi-Agent Systems. The programming language and project manager of the MOMAS architecture simplify the implementation of the final system.

Our process can be used for the specification of the virtual agents with a high-level of abstraction. These agents are clustered into an abstract layer that interfaces with servers in the network. To test our proposal we apply it to a surveying system in a 3D Chat. The different agents cooperate with each other to achieve the system's global goals. In our example, the goal is to survey the users about the university facilities in order to plan possible organizational changes.

In the future, the HABA process should be completed to all development phases. In particular, we must include the following stages: the stage for obtaining the requirements, and the stage for the verification and testing of the systems. We think that some established techniques of software engineering can be extended for that purpose. For the verification and testing stage, it is necessary to formalize the programming language and communications for the MOMAS technology. It can be made by means of denotational approach [30]. Moreover, it is necessary to improve the HABA.PM development tool in order to facilitate its extensive usage.

ACKNOWLEDGEMENTS

This work has been partially supported by grants TIN2007-68066-C04-02 of the Spanish Ministry of Science and Education and by FEDER Founds, also by grant P1-1B2007-56 of the Fundació Caixa Castelló – Bancaixa.

REFERENCES

[1] Mondéjar-Andreu, R. García-López, P., Pairot-Gavaldà, C., Gómez-Skarmeta, A. Tracking the Evolution of Collaborative Virtual Environments. UPGRADE, Vol. VII, No. 2, April 2006, pp. 24-29. 2006.

[2] Ibáñez Martínez, J., Delgado-Mata, C., Aylett, R. Virtual Environments: A Multi-disciplinary Field. UPGRADE, Vol. VII, No. 2, April 2006, pp. 2-4. 2006.

[3] Gehorsam, R. The Coming Revolution in Massively Multi-user Persistent Worlds. Computer 36 (4), pp 93-95. 2003.

[4] Zambonelli, F., Omicini, A. Challenges and Research Directions in Agent-Oriented Software Engineering. Autonomous Agents and Multi'agent Systems, Vol. 9, No. 3. 2004.

[5] Zambonelli, F., Jennings, N., and Wooldridge, M. Developing Multi-agent Systems: The Gaia Methodology. In ACM Transactions on Software Engineering Methodology, vol. 12, no.3, pp. 317-370. 2003.

[6] Kinny, D., Georgeff, and Rao, A. A methodology and modelling technique for systems of BDI agents. 7th Eureopean Workshop on Modeling Autonomous Agents in a Multi-agent World. LNAI vol. 1038, pp.56-71. Springer-Verlag. 1996.

[7] Iglesias, C.; Garijo, M.; González, J. and Velaso, J. Analysis and Design of multi-agent systems using MAS-CommonKADS. Intelligent Agents IV, LNAI vol. 1365, pp. 313-326. Springer Verlag. 1998.

[8] DeLoach, Scott A. Analysis and Design using MaSE and agentTool Proceedings of the 12th Midwest Artificial Intelligence and Cognitive Science Conference (MAICS 2001). Miami University, Oxford, Ohio, March 31 - April 1. 2001

[9] GRASIA Group (2009), INGENIAS (Available in http://grasia.fdi.ucm.es/ingenias/).

[10] Wooldridge M., Jennings, N, and Kinny, D. The Gaia Methodology for Agent-Oriented Analysis and Design. Autonomous Agents and Multi-Agent Systems, Vol. 3, No. 3, September, pp 285-312. 2000.

[11] Iglesias, C., Garijo, M. and González, J.C. A survey of Agent-Oriented Methodologies. In: Muller, J.P., Singh, M., and Roa, A.S. (Eds.), Intelligent Agent V, Proceeding of ATAL-98, Springer, LNCS 1555, pp. 317-330. 1999.

[12] Cernuzzi, L. and Rossi, G. On the Evaluation of Agent Oriented Methodologies. Proceedings of the OOPSLA 02 - Workshop on Agent-oriented Methodologies, November 2002, Seattle (USA), pp. 21-30. 2002.

[13] Cernucci, L., Cossentino, M., and Zambonelli, F. Process models for Agent-based Development. Journal of Pervasive and Mobile Computing, Vol. 18, No. 2. 2005.

[14] Winikoff, M. Future Directions for Agent-Based Software Engineering. International Journal of Agent-Oriented Software Engineering. Volume 3, Issue 4, pp. 402-410. 2009.

[15] Barella, A., Carrascosa, Botti, V. "JGOMAS: GameOriented MultiAgent System based on Jade". ACM SIGCHI International Conference on Advances in Computer Entertainment Technology. 2006.

[16] Davies, N.P., Mehdi, Q.H., Gough, N.E, Anderson, D., Jacobia, D. & Bornes, V.V. A review of potential techniques for the creation of intelligent agents in virtual environments, Proc. 5th Int. Computer Games Conf. CGAIDE'2004, Microsoft Reading UK, pp 248-256. 2004.

[17] Laird, J., van Lent, M. Human-Level AIs Killer Application. AI Magazine, 15–25 (summer, 2001). 2001.

[18] Nareyek, A. Intelligent Agents for Computer Games. In: Marsland, T., Frank, I. (eds.) CG 2001. LNCS, vol. 2063, pp. 414–422. Springer. 2002.

[19] Niederberger, C., Gross, M.H. Towards a Game Agent. Institute of Visual Computing, ETH Zürich, Technical Report 377. 2002.

[20] Aranda, G., Carrascosa, C. and Botti, V. Intelligent agents in serious games. In Fifth European Workshop on Multi-Agent Systems (EUMAS 2007). Association Tunnisienne dIntelligence Artificielle, 2007.

[21] Argente, E., Palanca, J., Aranda, G., Julian, V., Botti, V., Garcia-Fornes, A., Espinosa, A. Supporting agent organizations. In: Burkhard, H.-D., Lindemann, G., Verbrugge, R., Varga, L.Z. (eds.) CEEMAS 2007. LNCS (LNAI), vol. 4696. Springer, Heidelberg, pp. 236–245. 2007.

[22] Aranda, G., Carrascosa, C. and Botti, V. Characterizing Massively Multiplayer Online Games as Multi-Agent Systems. E. Corchado, A. Abraham, and W. Pedrycz (Eds.): Hybrid Artificial Intelligence Systems (HAIS 2008), LNAI 5271, pp. 507–514. 2008.

[23] Cernuzzi, L., Juan, T., Sterling, L., and Zambonelli, F. The Gaia Methodology: Basic Concepts and Extensions. In Methodologies and Software Engineering for Agent Systems. Kluwer. 2004.

[24] Akbari, Z. O., and Faraahi A. Evaluation Framework for Agent-Oriented Methodologies. Proceedings of World Academy of Science, Engineering and Technology, Vol. 35 nov. 2008, pp. 419-424. 2008.

[25] Garcés, A., Ricardo Quirós, Miguel Chover y Emilio Camahort. Implementing Moderately Open Agent-Based Systems. Proceedings of IADIS International Conference WWW/Internet 2006. Murcia, España. 5-8 Octubre 2006.

[26] Garcés, A., Ricardo Quirós, Miguel Chover, Joaquín Huerta, Emilio Camahort. Building Moderately Open Multi-Agent Systems: The HABA Process. Lecture Notes in Electrical

Engineering , Vol. 27. Proceedings of the European Computing Conference, Springer-Verlag, pp. 337- 345, 2009.

[27] Garcés, A., Ricardo Quirós, Miguel Chover, Joaquín Huerta y Emilio Camahort. A Development Methodology for Moderately Open Multi-Agent Systems. Proceedings of IASTED International Conference on Software Engineering (SE 2007). ACTA Press. 13-15 Febrero 2007.

[28] Aranda, G., Botti, V. and Carrascosa, C. MMOG based on MAS: The MMOG Layer. AAMAS 2009, 8th International Conference on Autonomous Agents and Multiagent Systems, 10–15 May, 2009, Budapest, Hungary, pp. 1148-1150. 2009.

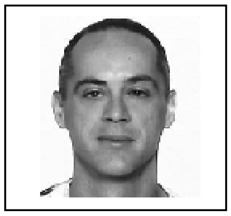
[29] Chover, M., Ó. Belmonte, I. Remolar, R. Quirós, J. Ribelles. Web-based Virtual Environments for Teaching. Proceedings of Eurographics/ACM SIGGRAPH Workshop on Computer Graphics Education, July 6-7 2002, Bristol University, UK. Especial issue of VIRtual, Electronic Journal on Visualization, Interactive Systems and Pattern Recognition, pp. 83-87, 2002.

[30]. Garcés, A., Ricardo Quirós and Miguel Chover. Global Communications in Moderately Open Multi-Agent Systems. Proceedings of IEEE International Conference of Intelligent Systems and Intelligent Computing, IEEE Society Press. Shanghai, China. 20-22 Noviembre 2009

Authors

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.

Dr. Ricardo Quirós is an Associate Professor at the Universitat Jaume I of Catellón. He received his PhD degree in Computer Science at the Technical University of Valencia in 1996. 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.



Miguel Chover received his PhD degree in Computer Science in 1996, from the Technical University of Valencia, Spain. He is Assistant Professor of Computer Science at Universitat Jaume I, Spain. He is member of the executive committee of Eurographics (Spanish Chapter). His research areas include multiresolution modeling, real-time visualization and virtual worlds.

Emilio Camahort is Associate Professor at the Technical University of Valencia, Spain, where he currently teaches Computer Graphics and Graphical User Interfaces. His research interests are in the areas of Computer Graphics and Interactive Techniques, Multi-modal Interfaces and Parallel and Distributed Computing. He is a member of ACM and Eurographics.