Introducing Intelligent Agents into Massively Multi-User Persistent Worlds

A. GARCÉS, R. QUIRÓS, M. CHOVER, J.HUERTA
Department of Computer Systems
Jaume I University, Castellón, SPAIN

E. CAMAHORT
Department of Computer Systems
Politechnic University of Valencia, Valencia,SPAIN

Abstract: - In this work we propose an architecture and methodology to add virtual agents to Massively Multi-User Persistent Worlds. It is based on a restricted class of Multi-Agent Systems that meets the requirements of the problem and simplify the implementation process. To demonstrate the application of our methodology we add 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.

Key-Words: - Virtual Environments, On-Line Games, Intelligence for Games, Agent Systems

1 Introduction

In recent years developments in gaming technology have focused on Massively Multi-User Persistent Worlds (MMPW). This virtual communities host thousands of users interacting with each other through the Internet in real time. Such communities require new distributed programming paradigms, as well as virtual life management. Persistence is also a much needed feature that implies a virtual world that changes continuously, even when the users are not connected. Social relationships prevail over years, and complex societies may live forever on line. Users are not the only living objects inside an MMPW. Non-player characters 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.

At a semantic level MMPW intelligence implementations are highly dependant on the application domain. In most cases intelligence has been specified using AI techniques embedded in the application. This prevents changing or removing faulty intelligence. Alternatively, agent-based techniques support a more natural specification of modular, distributed and cooperative systems. Their agents are autonomous components that can perceive the environment and react to it. Agents' social behavior allows them to interact with each other and form multi-agent societies.

Despite their advantages, agent-based systems have not been widely used in game programming. The reason is that they are too abstract and lack the tools needed to implement the system from the abstract specification.

To overcome these problems we propose an architecture and methodology to add virtual agents to MMPW. It is based on a class of Multi-Agent Systems (MAS) named Moderately Open Multi-Agent Systems (MOMAS). These systems impose a static organizational structure, their agents are homogeneous, and they have a centralized mechanism that manages the agents of the system. The MOMAS methodology has the advantage that it allows both high- and low-level behavior specifications. It comes with a programming language and a development framework for fast prototyping of MAS. This allows using modern programming techniques and tools common in other programming paradigms like OO programming. These techniques and tools reduce the gap between abstract specification and system implementation.

To demonstrate how our methodology can be applied to an MMPW, we add a set of autonomous virtual agents to a 3D chat. They survey the users for data that may be used to evaluate the processes and components of the virtual world.

Our paper is organized as follows. In the background section we describe MMWP and multi-agent systems. Section 3 introduces intelligent control of Non-Player Characters and the architecture of a MOMAS. In Section 4, we model a survey application that gathers information from the users of a 3D chat. Section 5 presents our implementation and

results. The last section includes our conclusions and directions for future work.

2 Background

The integration of AI techniques in MMPW has been outlined in several works, using different approaches. The most promising approach tries to add intelligent agents to the virtual world. This kind of agents is usually called Virtual Agents. In this section we survey previous work in MMPW design and Multi-Agent Systems.

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 [1]. AI technologies have been included in these MMPW using either formal approaches, or scripting approaches and basic data structure designs [2].

Formal AI technologies include Finite State Machines, Fuzzy State Machines and Neural Networks. Intelligence is programmed using algorithms embedded in the architecture of the Virtual Reality system. This is costly and prevents changes of the system's AI.

Alternatively, AI can be programmed using autonomous components that are simpler but use more complex communication protocols. These protocols allow more sophisticated cooperation tasks. Agent-based systems are one of these techniques that naturally support a large class of distributed and cooperative systems.

Different techniques have been proposed to build Multi-Agent Systems (MAS), including BDI [3], MAS-CommonKADS [4], and GAIA [5]. These are good abstractions, but they do not include specific programming languages and tools for the implementation of MAS. This problem has been partially solved in platforms like JADE [6], ZEUS [7] or AgentTool [8]. Unfortunately, they preserve many of the problems of their underlying programming languages, or they introduce unnecessary abstractions unsuited for practical implementation.

Despite these limitations, some practitioners apply MAS technology to implement intelligence in games or virtual environments. Barella et al include intelligence in computer games using the JADE platform [9]. Davies et al. use BDI to specify the behavior of Virtual Agents [10]. Finally, agent systems have been used to implement commercial games like The Sims (www.thesims.com) and Black and White (www.lionhead.co.uk). However, all these systems have limitations when representing agent social abilities.

3 Virtual Agents

Adding AI to Non-Player Characters (NPCs) , also called virtual agents, is the most important application of AI to MMPW. It allows creating more realistic environments and more immersive experiences.

3.1 Virtual Agent Features in MMPW

Virtual environments and virtual agents have the following features.

- A virtual environment is entirely observable, that is, all the sensors of any agent are capable of sensing the data relevant to the agent's decision making.
- Virtual agents are deterministic, that is, the next state of an agent is uniquely given by its current state and the action it is about to execute.
- The agent's behavior is sequential since decisions are made taking only into account previous decisions.
- Even though the virtual environment is dynamic, we assume that it does not change substantially while the agent is making a decision.
- The multi-agent system is cooperative, that is, it is made of multiple virtual agents that cooperate with each other to maximize their performance.

These features suggest certain agent models that reduce development and maintenance costs while keeping realistic and immersive environmental properties. In the following section we present an MAS subclass that is targeted at this kind of architecture.

3.2 Moderately Open Multi-Agent Systems

We describe how to organize agents in communities and how to model and architect a MOMAS. Our methodology allows modeling and prototyping MOMAS with the above architecture. Such systems run on homogenous distributed operating platforms, each associated with a server.

We assume that 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. It has tools for language interpretation, message routing and information management.

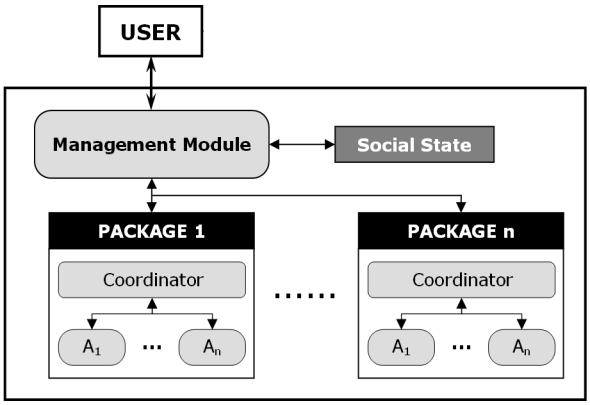

Fig. 1. Architecture of a Moderately Open Multi-Agent System

The development framework suitable for the MOMAS architecture includes a Development Methodology, a Programming Language and a Project Manager for fast prototyping. These systems have special features related to the global system environment and the management of information resources. We also introduce a notation that using a top-down approach improves the relationship between abstract analysis and design, and system implementation. We do not describe the entire development framework because it is beyond the scope of this paper. A description of the models included in the design and analysis of a MOMAS can be found in a paper by Garcés et al. [11]. Here we present a brief description together with a development example that uses this methodology.

4 Building an Intelligent Virtual Environment

We describe and demonstrate the use of our 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 textual questions that the users can freely answer in natural language.

In order to specify the system we use our MOMAS development methodology. This methodology has three fundamental elements: 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 whole development process, from the analysis to the implementation, is performed by refining the roles step by step.

Agents Classes are computational abstractions that define the common behavior of a set of entities. An Agent is an instance of an Agent Class with two states: a concrete state and an abstract state. The concrete state is dynamic and specifies the mental state of the agent. The abstract state is static and common to all the agents of a class.

Modeling the environment is another important issue in our methodology, since other MAS technologies do not support environmental models needed in MMPW. Environmental models support handling global resources and restrictions. The different components of the MOMAS interact with the resources of the environment and with each other. These interactions follow always the restrictions imposed by the system.

4.1 The Social Object Model

The social object model contains the computational resources of the MAS. It identifies the objects of the environment and their general restrictions. In our 3D chat, the social object model is an abstract representation of the MMPW scene. Figure 2 shows the social object model of our system expressed in Z-notation. $\aleph$ denotes the set of natural numbers, $\Re$ denotes the set of real numbers, and $\Re^{[a,b]} = \{x \in \Re \mid a \leq x \leq b\}$.

Social Objects : Survey System

Types
CHAR, POS, TIME
(*… Other types …*)

Objects
Region ≅ [id_Region : ℵ; p: POS; radius: ℜ]
Question ≅ [id_Question : ℵ; text: seqCHAR]
Answer ≅ [id_Question: ℵ; t: TIME; id_Region: ℵ1; id_Avatar: ℵ1; resp: seqCHAR]
Position ≅ [id_Avatar : ℵ; p: POS]
Efectiveness ≅ [id_Region; effectiveness: $\Re^{[0,1]}$]
(*…. Other Objects ……*)

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

Fig. 2. Social object model of the 3D Chat

The environment contains the positions of the users and the virtual agents, the set of questions of each survey, and the subdivision of the environment into regions. We register information for each region to

determine its effectiveness for each survey. The effectiveness of the regions is used to manage the movements of the surveyors. The model imposes a constraint to guarantee that the identifier of each object is unique.

4.2 The Role Model

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. In this section we specify some of these roles using our MOMAS methodology.

The Observer role combines the positions of the users and the avatars in the scene with the effectiveness of the regions to determine an optimal position to start a survey. It sends this optimal position as a message to the surveyor and waits for surveyor acknowledgment. The Observer role can update the Social State and requires access to the effectiveness value of each region. Figure 3 shows the specification of the Observer role model using our methodology.

Role Observer

Description
Tracks the scene and the users to inform Surveyors.
Services
InformEvalMovement [Surveyor] : POS → Boolean
Activities
GetPosition : → Boolean
SetPOS : → POS X ℵ
Wait : → Boolean
Permissions
changes Position
reads Region, Effectiveness
Responsibilities
Liveness
Observer = (Getposition( ). SetPOS(out p: POS; out n: ℵ).
Surveyor:: SuggestMovement(p, n).
InformEvalMovement(in p1: POS). Wait( ))*
Safety True

Fig. 3. Observer role model

The goal of the Surveyor role is to survey the users of the 3D Chat. It waits for a movement recommendation from the observer, evaluates the recommendation using its own knowledge and informs its decision to the observer. After this, it moves to the optimal position to survey the users. The surveying process randomly selects questions and collects answers from the users. The Surveyor role modifies the Answer object and needs access to the Question, Region and Effectiveness objects. Its safety property establishes the maximum time it will wait for the users to answer. Figure 4 shows the specification of the Surveyor role model using our methodology.

In our example, the Analyst role only determines the effectiveness of each region. More sophisticated tasks such as processing the answers in natural language or applying changes to the organization of the MMPW can be added using the proposed methodology.

Role Surveyor

Description
Survey the users.
Services
SuggestMovement [Observer] : POS X ℵ → Boolean
Activities
EvaluateMovement : POS X ℵ → 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: ℵ).
EvaluateMovement(in p: POS; in n: ℵ; out p1: POS).
Observer:: InformEvalMovement (p1).
Move(in p1: POS).
(SelectQuestion(out q: Question).
SendQuestion(in q: Question))+.
(AwaitAnswer( ) | WaitRegión( ))*
Safety $(t_a - t_0) < t_e$

Fig. 4. Surveyor role model

5 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).

The surveying module has been modeled with our MOMAS methodology, and has been implemented

using our own programming language and project manager. These three tools form a development framework that allows fast prototyping of a MOMAS specification. Figure 5 shows a screenshot of our development tool. The left-hand side of the window contains the system's main components organized in a tree. They include packages, agent classes, roles and a console program that support batch execution. When a component is selected its associated editor runs on the Editor window. The console is used to prototype and debug the MOMAS, and launch its execution. The application can be compiled to a program written in JAVA or C++.

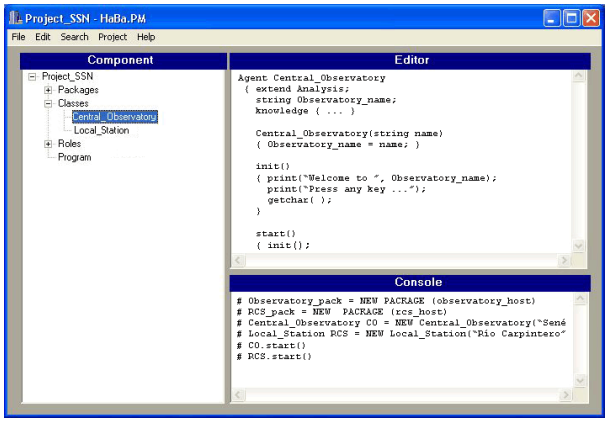

Fig. 5. MOMAS development tool

Fig. 6. A screenshot of the final application: a surveyor approaches a group of users

Fig. 7. Another screenshot of the final application: performing a survey

Figures 6 and 7 show two screenshots of our final application. In Figure 6 a surveyor moves towards a group of users to start a survey. In Figure 7 we can see how the survey is performed, with the surveyor asking the users their opinion about the library.

4 Conclusions and Future Work

We have presented a development framework for fast prototyping of MOMAS. Our methodology can be used to add intelligent agents to an MMPW. The main advantage of the proposal is that it allows both high- and low-level specification of Multi-Agent Systems. Our MOMAS methodology supports the specification of the virtual agents with a high-level of abstraction. The programming language and project manager of the MOMAS architecture simplify the implementation of the final system, reducing costs and improving future system developments.

To test our proposal we apply it to a surveying system in a 3D Chat. The system includes three different kinds of agents: observers, surveyors and analysts. 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 our example system we did not develop the Analyst role. We expect to add this functionality to analyze the results of the surveys and make decisions accordingly. For example, we may want to change the university's environment to meet the users' expectations, adding shops or changing the size of the classrooms. Or we may want to modify the

Surveyor's behavior to ask more specific questions or change the philosophy of the surveys.

References:

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

[2] Brian Schwab "AI Game Engine Programming". Ed. Charles River Media. ISBN: 1584503440. 2004.

[3] 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

[4] Iglesias, C.; Garito, 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

[5] 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 2000, pp 285-312

[6] Ballifemine, F., Poggi, A. and Rimassa, G. JADE: a FIPA2000 compliant agent develop-ment enviroment. Proceedings of the fifth international conference on Autonomous Agents, ACM, 2001

[7] Nwana, H. S., Ndumu, D.T., Lee, L.C. and Collis, J.C. ZEUS: A Toolkit for building Dis-tributed Multi-agent Systems. Applied Artificial Intelligence Journal, vol.1, No.13, pp.129-185, 1999

[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] Barella, A., Carrascosa, ., Botti, V. "JGOMAS: GameOriented MultiAgent System based on Jade". ACM SIGCHI international conference on Advances in computer entertainment technology. ISBN: 1-59593-380-8. 2006

[10] 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, ISBN 0-9549016-0-6, 2004

[11] A. Garcés, R. Quirós, M. Chover, J. Huerta and E. Camahort, "A Development Methodology for Moderately Open Multi-Agent Systems", IASTED Conference on Software Engineering, Innsbruk, Austria, February 2007.

.