Building Quests for Online Games with Virtual Institutions

G. Aranda[1], T. Trescak[2], M. Esteva[2], and C. Carrascosa[1]

[1] Universidad Politécnica de Valencia
Departamento de Sistemas Informáticos y Computación
Camino de Vera, s/n – 46022 Valencia – Spain
{garanda, carrasco}@dsic.upv.es
[2] Artificial Intelligence Research Institute (IIIA)
Spanish Scientific Research Council (CSIC)
Campus Universitat Autonoma de Barcelona
08193 Bellaterra, Catalonia, Spain
tomi.trescak@gmail.com, marc@iiia.csic.es

Abstract. This document describes how to re-purpose an existing agent technology called Virtual Institutions as a mechanism to define new "quest" elements in Massively Multiplayer Online Games based on Multi-Agent Systems. Quests are a very important part of most Massive Online Games as they wield to flow and narrative of the game in a linear or non-linear manner.

1 Introduction

Massively Multiplayer Online Games (MMOGs) are an important focus of research, not only because they are economically attractive, but also because a MMOG involves many fields and a large amount of data that is generated by the interactions of many individuals: configuring a MMOG is a relevant source of research. In the field of AI, to model such systems and its dynamics is nowadays a very relevant task[17, 11].

Massively Multiplayer Online Games, by nature, are played "in the cloud", i.e. in a virtual world away from the players' computers and game devices that are hosted in a large array of servers. The complex and distributed nature of these kind of games makes them impossible to be played relying just in the players' hardware and resources, as some other types of online games do. This distributed nature is also one of the many parallel factors between MMOGs and Multi-Agent Systems, as it has been described in some previous works[1, 3].

Also, MMOGs usually spot a very open-ended nature and narrative, basically allowing the players to roam the virtual game world free doing largely whatever they want to do. However, these games also use quite often the concept of **quest** or instance: A quest is a specific mission designed to be fulfilled by the players of a game. It is played in a different flow than the open-ended virtual game world, a more straightforward and linear flow, similar to the flow of classic offline games. This mission may include the involvement of other characters in the game

(corresponding to players or not) and different sequences of actions to fulfill in different places in the game.

For example, suppose a player is playing a science fiction game (similar to the well-known "Star Wars Galaxies"™or "Star Trek Online"™games) in which players may travel freely with a spaceship through the cosmos. Players may encounter another character in the game, a computer-controlled character called the "quest-giver" which gives them the opportunity to embark on a singular quest looking for a specific item on a planet. The moment the players accept this endeavor, a new linear narrative opens just for them (and their potential companions), and a new set of sequential goals and rewards becomes available for the players.

This paper presents a new addition to the existing architecture of *MMOG based on Multi-Agent Systems*[3]: the making of quests using Virtual Institutions[8], detailing, not only the architecture (ontology and agent taxonomy), but also a prototype applied to a concrete game example.

In section 2 Virtual Institutions are presented. Later, in section 3 MMOG based on MAS architecture is also presented. Following, the concept of quest in this kind of systems is introduced in section 4. Additionally, section 5 explains how to develop these quests using VI technology. Finally, some conclusions and future lines of work are presented in section 6.

2 Virtual Institutions

Virtual Institutions are a 3D Virtual Worlds with normative regulation of interactions [8]. This concept appeared as a combination of electronic institutions [12] and 3D virtual worlds. In this context, electronic institutions are used to specify the rules that govern participants' behaviors, while 3D virtual worlds are used to facilitate human participation in the institution. The design of Virtual Institutions is divided in two separate steps: i) specification of the institutional rules, and ii) generation of the virtual world.

The institutional rules are defined using Electronic Institution model composed by the following components:

- *Dialogical Framework*. It defines a common ontology and communication language to allow humans with different cultural backgrounds, as well as, agents to exchange knowledge. This ontology and language for humans will be further transformed into actions that are allowed to be executed in the Virtual World. Those actions are connected to 3D models in the environment, the affordances of which will help in eliminating the cultural barrier. Due to the further provided translation of the communication language into actions and vice-versa, the agents will be able to interact with humans and understand their actions. The dialogical framework also fixes the organizational structure of the society, that is, which roles can participants play, and relationships among them.
- *Scene*. Interactions between agents in order to jointly perform an activity are articulated through agent group meetings, which we call *scenes*, with a

well-defined communication protocol. We consider the protocol of a scene to be the specification of the possible dialogues agents may have. Hence, a scene is specified as a deterministic finite automata, whose states represent interaction states, while arcs are labelled with messages (illocutions) or timeouts. Notice however that the communication protocol defining the possible interactions within a scene is role-based instead of agent-based. In other words, a scene defines a role-based framework of interaction for agents.
- *Performative Structure.* Scenes can be connected, composing a network of scenes, that we call performative structure, to capture the relationships among scenes. The specification of a performative structure contains a description of how agents can legally move from scene to scene by defining both the pre-conditions to join and leave scenes. Satisfying such conditions will fundamentally depend on the roles allowed to be played by each agent and its acquired commitments. The execution of a performative structure equates to the execution of the multiple, possibly simultaneous, ongoing activities, represented by scenes, along with the agents participating in each activity. Agents within a performative structure may be possibly participating in different scenes, with different roles, and at the same time.
- *Norms.* They determine the consequences of user actions. These consequences are modeled as commitments that participants acquire as a consequence of their actions and have to fulfill later on. These commitments may restrict future activities of the users.

Furthermore, for each role activity and performative structure the designer can define an information model, which is composed of a set of attributes that will be used to keep the state of the agent or an activity. For instance, the information model of a player can contain its credit, points, or the objects it have. The values of such attributes are modified depending to the evolution of the institution. That is, when an action is executed some of the attributes are modified. For instance, the points of a player can be increased after successfully completing a quest.

Once the institutional rules have been specified it is time to generate the virtual world. This can be automatically done taking into account the activities can engage on defined in the specification. Specifically, a 3D room is represented for each activity (scene). As a result a mapping is created between the activities defined in the specification, and where these activities occur within the virtual world. In addition, messages specified in scene protocols are mapped to actions supported by the virtual world. For instance, in the context of an auction house raising a hand can be mapped to the message for submitting a bid.

In contrast to Electronic Institutions, the normative part of a Virtual Institution does not represent all the actions that are allowed to be performed in a Virtual World. Specifically, those actions that require institutional verification are those mapped to scene messages. The rest of the actions provided by the virtual world software can be freely executed. The specification of the institutional rules can be regarded as valid sequences of actions among the ones that require institutional verification. In addition, the attributes associated to the

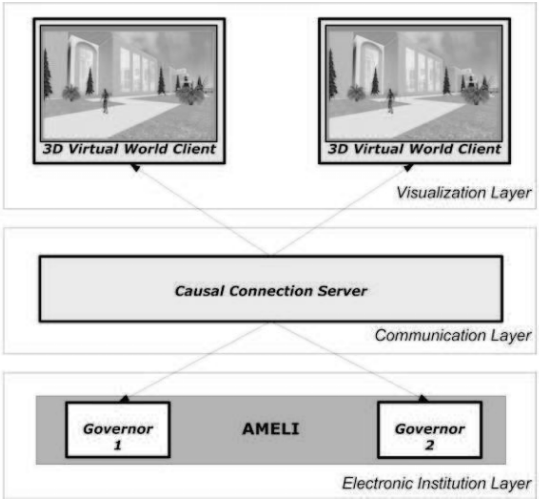

Fig. 1. Runtime Architecture.

different roles, and activities help to keep participants or activities state. They keep information of past actions which are relevant to determine the validity of future actions.

Virtual Institutions are deployed by a 3-layered infrastructure presented in Figure 1.

First layer is the *Electronic Institution Layer*. It uses the AMELI system [4] for enforcing the institutional rules established on the specification step. AMELI keeps the execution state of the institution and uses it along with the specification to guarantee that participants' actions do not violate any of the institutional constraints.

Second layer is the *Communication Layer*. Its task is to causally connect [16] the institutional infrastructure with the visualization system transforming the actions of the visualization system into the messages, understandable by the institutional infrastructure and the other way around. This causal connection is done via the Causal Connection Server using the mapping between institutional messages and virtual world actions. The causal connection is happening in the following way: an action executed in the 3D Virtual World (that requires institutional verification) results in a change of the institutional state in the AMELI layer, as well as every change of the institutional state is reflected onto the 3D Virtual World and changes its state. The Communication layer conceptually and technologically connects two metaphors: Electronic Institutions and Virtual Worlds and we see it as one of our major scientific contributions.

The third layer is called *Visualization Layer*. It is used to visualize the 3D Virtual World for the users.

3 Massively Multiplayer Online Games based on MAS

Deploying a game like a MMOG is like deploying a major software project to act as a service in the cloud. It presents all the issues and hazards one could expect from deploying a large software system to solve a big and distributed problem: need for good scalability, distribution of knowledge, user load balance, network bottlenecks, long development cycle and asynchronous events, just to name a few.

There have been other approaches to use agents in online gaming, albeit not exactly in the MMOG space. Most of them are oriented towards achieving better behaviors in Non-Player Characters. Dignum et al. [10] propose a more natural (long-term) behavior of Non-Player Characters through the use of Multi-Agent Systems, and clarify that game design should be adjusted to incorporate the possibilities of agents early on in the process, a statement also fundamental to this line of research. Also, Gemrot et al. [15] take a more practical approach by developing a full framework, called *Pogamut*, to integrate distributed intelligent agents as synthetic opponents and allies (bots) in games powered by the "Unreal Engine"^{TM}technology. The main objective of the *Pogamut* project is to provide new AI-driven players that can bestow new challenges to the players and learn from their actions, using a distributed AI network that runs outside of the game clients and server. Both approaches take online gaming in general as a domain for agents achieving good results, so it is a natural step forward for agent technology to enter the MMOG space.

3.1 Architecture

A MMOG (like most complex systems) can be seen as a system split into several layered subsystems, with each layer being relatively independent and taking care of one aspect of the whole MMOG experience. From the perspective of this work, a MMOG is split into three layers:

HCI Layer: It is the *client-side* of the system, the part of the game running on the players hardware (PC, mobile phone, game console, ...). It is the user interface that the game provides to the player, i.e. the game *client* software, and it provides the player with a gaming experience (i.e. 3D graphics, sound...). It is the framework of the *InterfaceAgent* [2].

Intelligent Virtual Environment (IVE) Layer: It is the virtual representation of the game environment itself. It is part of the *server-side* of the system, the part of the game that runs mostly on the game provider's hardware, a controlled environment. The synthetic *place* and scenario where the game takes place: the virtual world. This world is independent of the type of game or simulation it must give support. Also, the IVE is designed following an agent-based approach, so it can be seen as Multi-Agent System embedded into another, larger, Multi-Agent System. The IVE layer is thoroughly described in [6].

MMOG Layer: It is a complex subsystem where all the game logics and mechanics are implemented and must be solved at run-time. It operates in conjunction with the IVE layer, but it is not dependent of that subsystem. It im-

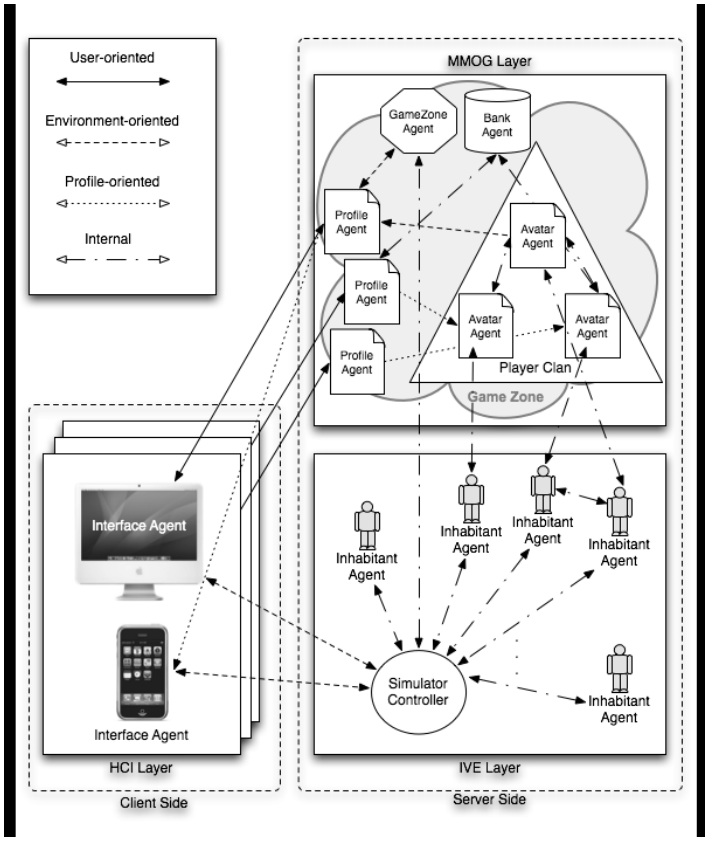

Fig. 2. The architecture of a MMOG based on MAS

plements the game rules / norms controlling the game development. It is the *place* where all the game clients connect to play and, along with the IVE layer, it must facilitate game server scalability. In this line of research, this subsystem is seen as the core of the MAS and requires, at least, one agent platform as its foundation. The MMOG Layer is the place that groups all the elements from the game which are independent both from the user space and the IVE, and thus is the core piece of the whole system, as it is where the actual game takes place. Section 3.2 describes this layer in more detail.

3.2 The MMOG Layer

As stated before, the **MMOG Layer** is essentially a dedicated, open MAS which runs the game. This MAS uses agent technologies like agent services, Electronic Institutions[13, 7] and Agent Organizations[14, 9, 5] to model some game mechanics, and translates the common issues and situations found in MMOGs into

problems that can be solved using classic software agent features, such as agent interactions, agent communication protocols (like auctions or call-for-proposals), service-oriented computing, event-driven behaviors or role models.

Like any other agentification process, one of the key ideas is to identify the agents and types of agents that will conform the system. In this case, the agents are based on the concepts and entities that form the whole game experience of a MMOG, and are explained in more detail in [2]:

ProfileAgent: a personal agent which manages the player status and profile within the game community. It manages the player's preferences in the game world, which avatars the player uses and the **role** that the player plays in the system (*Spectator*, *Player* or *GameMaster*).

AvatarAgent: an agent which represents an avatar of a human player within the game (a PC or Player Character). It is a persistent kind of agent: is not deleted and re-spawned often, it bears a long life cycle. It is the agent that holds the PC *stats* (server-side), and so, a malicious player cannot modify them locally (cheat). The AvatarAgent is the kind of agent that actually performs the actions for the player in the virtual world.

NPCAvatarAgent: an agent which represents an avatar of an AI-controlled character. It is similar to the AvatarAgent, as both populate the game world, but it does not obey nor represent a player in the game.

GameZoneAgent: a kind of agent which implements the logics of the game environment and works as a nexus between the *MMOG Layer* and the IVE Layer's Simulation Controller (see figure 2).

4 Defining Quests for MMOG

Quests are a very important part of most Massive Online Games as they wield to flow the narrative of the game in a linear or non-linear manner. Quests present the players with the opportunity to improve their virtual characters and their playing experience by grouping together players with the same objectives and guiding them through a segment of the overall game experience, rewarding players for their performance in the game. Quests also offer the designers of these open-ended games, an opportunity to develop more narrow-focused "levels", similar to those found in traditional offline games, without sacrificing the social aspects of online play and the overall goals of the game. Alas, the whole "career" of a virtual character can be seen as a series of linked quests towards an open-ended conclusion.

From the grand perspective of the game as a big Multi-Agent application, quests are a part of the system. They are seen as smaller Electronic Institutions with a semi-linear flow of the agents where custom and more strict norms exist than those from outside (those in the "open" areas of play). However, quests present some particular aspects that are not seen in other types of institutions.

Let's present the concept of "sub-quest" in the context of quests, which has a lot in common with the concept of scene in Electronic Institutions. A sub-quest

is one of the steps that compose a quest. A sub-quest can be defined as a set $S = (G, O, L, P, E)$, where:

- **G**: is the set of objectives (goals) player agents (P) must complete in the sub-quest. These objectives are expressed in the semantic ontology of the game itself using the OWL-DL language, which in turn is based on "MMOG Ontology" [3][3]. This set of goals cannot be empty, there must be at least one goal to play in the sub-quest.
  Goals are defined using the properties and data types present in the ontology, and usually make reference to modifying the state or the properties of an object of the game. For instance, let's presume a game uses an ontology in which the game characters (i.e. Avatars) have a numeric "health" stat. When a designer wants to create a goal that means "kill this enemy", it can be defined by declaring the "health" stat of the instance of NPCAvatar that represents that enemy to zero:

  ```
  Goal0: Avatar MyEnemy.health == 0.0
  ```

  Another example: let's say a designer wishes to express a goal that means "take this chest to the vault", it can be expressed by the *placedAtZone* property of the *chest* item:

  ```
  Goal1: GameItem MyChest.placedAtZone == "Vault"
  ```

  Initially, goals should not need to be decomposed into subgoals, but this is a feature that may be added in future iterations of this work.
- **O**: is the set of system agents (opponents) that oppose player agents (P) and keep them from completing the sub-quest by a process of conflict. Their view is the opposite of the player agents' view and its actions counter those of the players. However, contrary to classic game theory, these agents do not seek the Nash equilibrium [18] in the system of the quest. They seek to maintain the goals (G) in the initial state of the sub-quest, that is, without being fulfilled. This may be an empty set (i.e. no opponents).
- **L**: is the set of virtual locations (also called *Dungeons* and *Game Zones* in the literature) that serve as the environment around players (P) and the opponents (O). They are comparable to the rooms and transitions present in Electronic Institutions as they each have a maximum (and a possible minimum) concurrent agents of a particular type and in each one of the locations changes according to an interaction protocol, although the same protocol can be shared by more than one location. This may be an empty set, meaning that the sub-quest is not tied to a specific location.
  Locations can be specified using lists of derivatives of the *GameZone* and *GameBeacon* classes of the "MMOG Ontology". The designers of the game are expected to define and tag the virtual places where the game takes place using instances of those classes (or some sub-classes). Therefore, in practice, each L set will be a sequence of some of those instances, and the same instance may appear in different sub-quests (or quests) for different purposes.

[3] http://gti-ia.dsic.upv.es/ontologies/mmog.owl

- **P**: is the set of agents who play the role of players. These agents are usually controlled by human players that are playing the game. Sometimes an agent of the group P can be controlled by the platform, but it is rare. The aims of these agents are twofold: make sure the goals of the scene (G) are fully accomplished and maximize the profits (E) during the process. Ideally, an organization of player agents (P) wishes to complete all of the objectives of a quest and to get the maximum number of potential profits. Player Agents (P) are the antagonists of the Opponent Agents (O). This set cannot be empty, there must be at least one agent that plays the sub-quest.
- **E**: the function of earnings player agents (P) can obtain during the quest. In each sub-quest, this function changes to reflect the ratio of risk / profit present. Based on this ratio, the quests can be dimensioned. For example, a quest with a very large O component and a very low E component is seen as a "High Risk and Low Gain" quest and is less desirable for the players than a quest with a higher E component and lower O component, which would be "Middle Risk and High Gain". For a quest that has no earnings, this function can be equal to zero.

So then, a quest can be defined as a non-linear (or non-deterministic) sequence of connected sub-quests linked through outcomes. These outcomes are determined by the completion (or failing) of the goals in each sub-quest. Specifically, a quest can be seen as a directed acyclic graph where the nodes are sub-quests (as can be seen in figure 3). The designers of the quest must also specify which sub-quest (or sub-quests) serve as an entry point to the quest and which sub-quest (or sub-quests) serve as an exit point (i.e. end the quest). So, in the end, a quest can be defined in a very similar fashion to a non-deterministic finite automata:

$$Q = (SS, OC, I, F)$$

- **SS**: The set of sub-quests that compose the quest. The nodes of the graph.
- **OC**: The outcomes that connect the sub-quests. The strings of the graph.
- **I**: The subset of SS that are starting sub-quests for the quest.
- **F**: The subset of SS that are ending sub-quests for the quest.

Quests are semi-linear structures in nature. That means that a quest can follow a straight path from beginning to end, or that it can branch its path one or more times through its course. Nevertheless, every path that the players take will eventually lead them to ending the quest in one of the ending sub-quests of the quest. This gives the designers of the quest some sort of "elasticity" towards the development and narrative of the quest, as well as the ability to add some interesting gameplay elements.

For instance, suppose that a quest involves the players getting inside a locked room to retrieve an object (i.e. a treasure). The designer can create a sub-quest located in a contiguous room where an opponent (which has the door key) is guarding the entrance to the room. The designer may also create another sub-quest with no opponents that takes place in a backyard where a window leads

to the important room. Both sub-quests have similar objectives (i.e. get inside the important room), but they are presented as a choice to the players: Will the players take the indoor or outdoor path? If they take the indoor path, will they fight the opponent to get the key by force or will they try to bribe or sweet-talk the guard into give them the key? If they take the outdoor path, do they have the ability to open and jump off the window from the outside into the room? These are all gameplay choices that the developer may present to the players and implement them as branching paths in the quest design, but in the end, all of these choices lead to the same conclusion: the achievement of the ultimate quest goal and the completion of the quest.

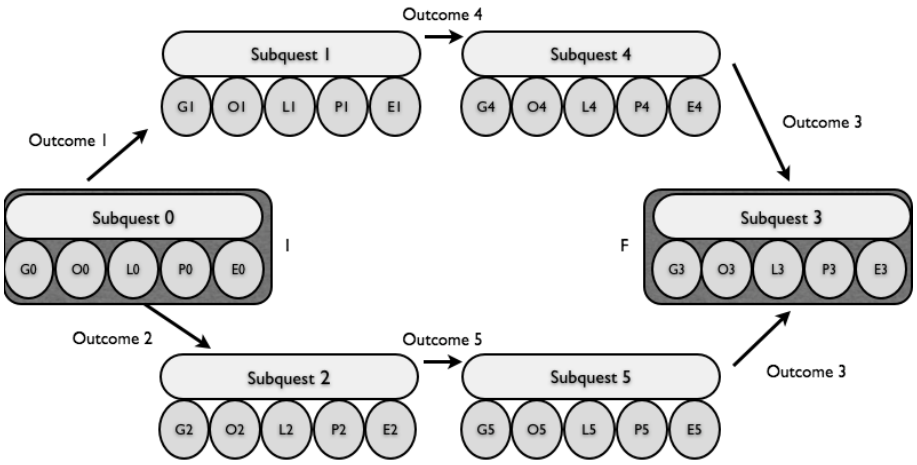

Fig. 3. Components of a quest

5 Building Quests with VIs

In this section the use of the VI framework and tools to develop quests for MMOG based on MAS is explained (specially using the Islander[12] editor). The initial approach followed in this work takes a formal specification of a quest and expresses it in practice using a VI. The transformation is implemented using the Islander tool from the Electronic Institutions Development Environment [4]. The Islander tool was developed as a user-friendly graphic interface for specifying Electronic and Virtual Institutions. In this work, it has been used in the first steps of developing quests for MMOG based on MAS (what is usually called the development *pipeline* in the videogame industry). In order to successfully implement a quest, the following points need to be taken care of: be able to express the knowledge using an ontology (this basically includes goals and earnings); be

able to control the flow of the agents participating in the quest and the roles they play (this basically includes players and opponents); be able to specify the flow and connections of the sub-quests; be able to express the different outcomes of a sub-quest; and be able to specify the sub-quests themselves.

Regarding the use of an ontology, the Islander tool allows the definition of custom ontologies with classes and properties, like the MMOG Ontology, that are used in the definition of the Performatives Structures and Dialogic Frameworks. So this need is fulfilled.

Regarding the agents, VI provide a unique type of internal agents called the Governors. These Governors are paired with the agents participating in the institutions and act as their real interface towards the system, preventing them from executing illegal actions based of their roles and characteristics (or stats, in MMOG terms). Besides, Governors "move" the agents through the scenes and transitions of the institutions as they interface with them. So this need is fulfilled.

Regarding the flow and connections of the quest, it's worth noting that the main element of a VI is its Performative Structure which, as seen in section 2, is essentially a directed graph that connects scenes through transitions. Although they are different concepts, a Performative Structure and a quest share the same kind of graphic representation. In fact, a quest may be seen as a subset of a whole Performative Structure, which contains all the possible sub-quests and all the possible paths that any agent can follow through the quest at any given time (or attempt).

Regarding the possible outcomes of a sub-quest, VIs use diagrams called *"Protocols"* to define the inner workings of a scene. Protocols are essentially simplified nondeterministic finite automata, with an arbitrary number of starting and final states, and where the state changes are triggered by common interactions between the agents of the scene or timed events. So, the different outcomes of a scene can be directly mapped to the final states of the automaton. A scene will have at least as many outcomes as final states has its inner Protocol (since more that one final state can lead to the same outcome). So this need is fulfilled.

Regarding the sub-quests, the goals (G), opponents (O), players (P) and earnings (E) have already been explained, as well as the outcomes. The location is the missing item. Locations are normally defined logically through the use of the MMOG Ontology (or one of its derivatives) by using the *GameZone* and *GameBeacon* classes and their possible subclasses. In VI there is not a explicit "location" field associated with a scene of the Performative Structure. Fortunately, these scenes may have as many additional *properties* as needed. A property is a semantic "key-value" pair (which may be mandatory to define in each scene) where the "value" part is a semantic expression (or a list of semantic expressions). By creating a "location" property in each scene that needs locations, this need is partially fulfilled. The other half of the question is the connection between the VI location and the actual representation of the virtual place in the IVE layer. This is done through the Causal Connection Server (as

seen in figure 1), but this problem falls outside the scope of this work and will be explained in detail in future articles.

In order to better explain this process, an example quest has been developed. This example quest introduced for this work is depicted in figure 4 and takes place in a fictional *science-fiction* MMOG game similar to the already mentioned examples in section 1. This quest follows the flow of a deed in which the players must track down a treasure stolen by space pirates. Players first receive the information of the deed by a non-player character (an agent of the class NPCAvatarAgent), which plays the role of "quest-giver" prior to accepting the quest (and entering the VI). This information presents the problem to the players (*"The evil space pirates have stolen a treasure. Their last known location are the ruins of an ancient base, but they are rumored to be in the orbit of a nearby planet."*) and the players must decide whether or not to embark on the quest. If so, the players are faced with an immediate choice: they can go to the ruins to look for clues or they can go to the nearby planet to confront the pirates. These two different paths correspond with a gameplay choice that the game designers wish to present the players with: will they use subtlety and insight or will they use brute force and a direct approach? Either way, players will eventually learn the true location of the treasure (the drifting remains of an old ship) and proceed there to try to find it. If they succeed, all ends well and the players "win" the quest and retrieve the treasure. If they are not able to find it, the quest has a bitter end as the players "fail" the quest and obtain no treasure whatsoever.

This quest has three types of scene: *FightScene*, with a protocol designed for the fighting; *FindScene*, for searches and tracking treasure; and *QuestEndScene* to resolve the end of the quest. The protocols of the scenes are quite simple. For example, *FightScene* is described in figure 5.

Internally, when the players accept the quest their AvatarAgents enter the VI. And the different gameplay branching choices are represented by different sub-quests and paths in the quest definition, very similar to what can be seen in figure 3. When the agents enter the VI, the first choice is represented by a XOR transition (*t0*) which either leads them the "FindTrace" scene or to the "FightPirates" scene.

If the players choose the first path, their agents transition to the "FindTrace" scene. In this scene, the goal is to find the clue (or clues) to the real location of the treasure. When the clue is found, the scene ends and its outcome brings the players to the next transition (*t1*).

However, if the players chose to confront the pirates, their agents are lead to a scene called "FightPirates". This scene may end in two ways: the players beat the pirates and obtain the location of the treasure by force, or the pirates kill the players and escape with the treasure. This dichotomy is represented by two possible outcomes, both passing through the *t1* transition: the first one is the normal flow of the quest and leads to the "FindCargo" scene. The second one, terminates the quest and leads to the "QuestEndBad" scene (and later to the "Exit" scene) of the institution.

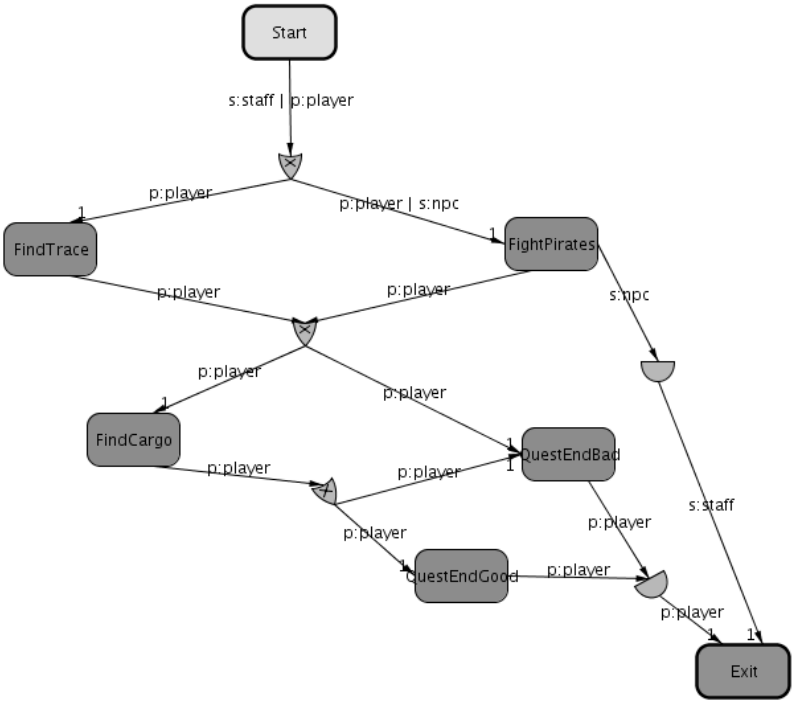

Fig. 4. Performative Structure of a quest as seen in Islander

When the agents arrive at the "FindCargo" scene, they have the chance to look for the hidden treasure, but they have to find it in a short amount of time. That limitation is represented using a timer in the protocol of the "FindCargo" scene. When the timer expires, the protocol ends with an outcome. However, if the players find the treasure, the protocol ends with a different outcome. If the players are unable to find the treasure, the quest ends in a negative way through the "QuestEndBad" scene. Nevertheless, if the players are able to find the treasure, the quest ends and the players receive a positive reward in their "QuestEndGood" scene. After that, the players' agents leave the institution and so the quest ends.

6 Conclusions and Future Work

In this work, a new addition to the existing architecture of *MMOG based on Multi-Agent Systems*[3] has been presented and successfully implemented: the making of quests using Virtual Institutions[8], detailing, not only the architecture (ontology and agent taxonomy), but also a prototype applied to a concrete game example.

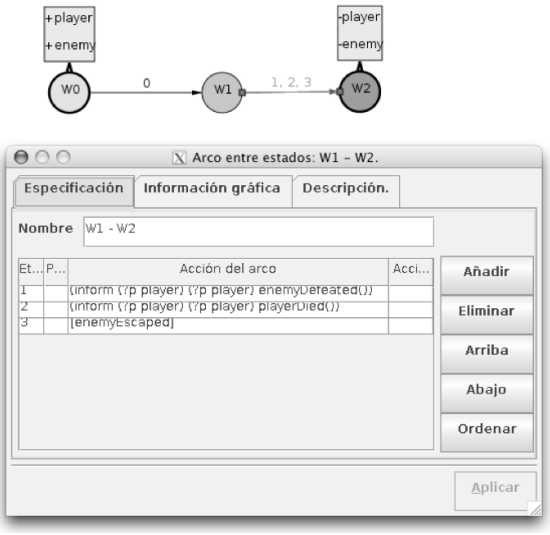

Fig. 5. FightScene Protocol

In the future of this line of work lie at least two new developments. The first one is to define all the possible interactions that can happen between agents populating a MMOG based on MAS and to integrate that knowledge into the definition of quests, in order to have better control and configuration of subquests based on agent interactions. The second one is to develop a methodology, addressed to game developers, to guide them in the use of this architecture for building their games.

7 Acknowledgements

This work has been partially funded by TIN2009-13839-C03-01, TIN2008-04446, PROMETEO/2008/051, GVPRE/2008/070 projects, CONSOLIDER-INGENIO 2010 under grant CSD2007-00022, project EVE (TIN2009-14702-C02-01), EU-FEDER funds, the Catalan Goverment (Grant 2005-SGR-00093) and Marc Esteva's Ramon y Cajal contract..

References

1. G. Aranda, V. Botti, and C. Carrascosa. Mmog based on mas: The mmog layer, (extended abstract). In *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, pages 1149–1150. Decker, Sichman, Sierra and Castelfranchi (eds.), 2009.
2. G. Aranda, C. Carrascosa, and V. Botti. 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, pages 507–514, 2008.
3. G. Aranda, C. Carrascosa, and V. Botti. *The MMOG Layer: MMOG based on MAS*, volume 5920, pages 63–78. Dignum, Bradshaw, Silverman, van Doesburg (eds.), 2009.
4. J. L. Arcos, M. Esteva, P. Noriega, J. A. Rodríguez-Aguilar, and C. Sierra. Engineering open environments with electronic institutions. *Journal on Engineering Applications of Artificial Intelligence*, 18(2):191–204, 2005.
5. E. Argente, J. Palanca, G. Aranda, V. Julian, V. Botti, A. Garcia-Fornes, and A. Espinosa. Supporting agent organizations. In *CEEMAS'07*, 2007.
6. A. Barella, C. Carrascosa, and V. Botti. Agent architectures for intelligent virtual environments. In *2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pages 532–535. IEEE, 2007.
7. A. Bogdanovych, H. Berger, S. Simoff, and C. Sierra. Narrowing the Gap between Humans and Agents in E-commerce: 3D Electronic Institutions. *K. Bauknecht, BP oll, and H. Werthner, editors, E-Commerce and Web Technologies, Proceedings of the 6th International Conference, EC-Web*, pages 128–137, 2005.
8. A. Bogdanovych, S. J. Simoff, and M. Esteva. Virtual institutions prototype. In *8th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009), Budapest, Hungary, May 10-15, 2009, Volume 2*, pages 1373–1374, 2009.
9. N. Criado, E. Argente, V. Julian, and V. Botti. Organizational services for spade agent platform. In *IWPAAMS07*, volume 1, pages 31–40. Universidad de Salamanca, 2007.
10. Dignum, F. and Westra, J. and van Doesburg, W.A. and Harbers M. Games and Agents: Designing Intelligent Gameplay. *International Journal of Computer Games Technology*, 2009, 2008.
11. N. Ducheneaut, N. Yee, E. Nickell, and R. Moore. Alone together?: exploring the social dynamics of massively multiplayer online games. *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 407–416, 2006.
12. M. Esteva. *Electronic Institutions: From Specification to Development*. PhD thesis, Artificial Intelligence Research Institute (IIIA-CSIC), Spain, 2003.
13. M. Esteva, B. Rosell, J. Rodriguez-Aguilar, and J. Arcos. AMELI: An Agent-Based Middleware for Electronic Institutions. *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 236–243, 2004.
14. E. Garcia, E. Argente, and A. Giret. Issues for organizational multiagent systems development. In *Sixth International Workshop From Agent Theory to Agent Implementation (AT2AI-6)*, 2008.
15. Jakub Gemrot, Rudolf Kadlec, Michal Bida, Ondrej Burkert, Radek Pibil, Jan Havlicek, Lukas Zemcak, Juraj Simlovic, Radim Vansa, Michal Stolba, Cyril Brom. Pogamut 3 Can Assist Developers in Building AI for Their Videogame Agents. *Proceedings of the First International Workshop on Agents for Games and Simulations*, 2009.
16. P. Maes and D. Nardi. *Meta-Level Architectures and Reflection*. Elsevier Science Inc., NY, USA, 1988.
17. M. Matskin. Scalable Agent-Based Simulation of Players in Massively Multiplayer Online Games. *Eighth Scandinavian Conference on Artificial Intelligence: SCAI'03*, 2003.
18. J. Nash. Non-cooperative games. *The Annals of Mathematics*, 54(2):286–295, 1951.