Multi-agent Modeling Methods for Massivley Multi-Player On-Line Role-Playing Games

Markus Schatten, Igor Tomičić, Bogdan Okreša Đurić
Artificial Intelligence Laboratory
Faculty of Organization and Informatics
University of Zagreb
Pavlinska 2, 42000 Varaždin, Croatia
Email: {markus.schatten, igor.tomicic, dokresa}@foi.hr

***Abstract*—Massively Multi-Player On-Line Role-Playing Games (MMORPG) give us the opportunity to study two important aspects of computing: (1) large-scale virtual social interaction of people (players) and (2) the design, development and coordination of large-scale distributed artificial intelligence (AI). A common denominator for both aspects are the methods used to study them: social interaction can be described and simulated using agent-based models (ABM) whilst distributed AI is commonly modeled in terms of multi-agent systems (MAS). The important question to ask in both perspectives is how do agents organize in order to perform their tasks and reach their objectives? In the following paper we will present and overview of agent-based approaches to modeling MMORPG's agents including human players, artificial players (bots) and non-playing characters (NPCs). We will put a special accent on useful agent behaviors, coordination and consensus mechanisms as well as organization and social network structure of agent societies.**

***Keywords*—MMORPG, multi-agent systems, modeling, npc, bot, agent behaviors.**

I. Introduction

Role-playing computer games (commonly referred to as role-playing games or RPGs) are a computer game genre in which the player controls the actions of a protagonist (or several party members) in some world which is well defined [1]. A massively multi-player on-line game (MMOG) is a computer game that supports a great number of players playing the game on-line simultaneously often causing interaction among them [2]. Massively multi-player on-line role playing games are thus a mixture of the RPG genre with MMO allowing players to control the action of their avatar (protagonist) by interacting with a potentially large user-base on-line [3].

MMORPGs have become enormously popular during the last two decades. According to Wikipedia [4] numbers of players on some popular MMORPGs are given below:

- Dofus – 10,000,000
- World of Warcraft – 7,600,000 (on average 34,000 simultaneously on-line)
- MapleStory – 5,000,000
- Knight Online – 4,250,000
- Start Trek Online – 3,200,000
- Guild Wars 2 – 3,000,000
- Final Fantasy XIV – 2,500,000 (avg. 30,000-45,000)

The global MMO market [5], [6] is growing continuously with 2011 $\approx$ 8.5 billion €, 2012 $\approx$ 10.2 Bn€, 2013 $\approx$ 11.7 Bn€ and 2014 $approx$ 15.0 Bn€.

While the importance of MMORPGs in economic terms is obvious, another aspect is of equal importance: it allows us to study two aspects of large-scale computing - social interaction of (large numbers of) players through a computing platform as well as the design and implementation large-scale distributed artificial intelligence (in form of non-player characters – NPCs, mobs – various monsters to be fought, as well as artificial players – bots). Both of this aspects can and should be studied using agent-based techniques the former by ABM (social science perspective of agents) and the latter by MAS (computer science perspective).

In the following work in progress paper we will present an overview of agent based modeling techniques which can be applied to study MMORPGs some of which have already been applied in the ModelMMORPG project[1]. Basically we will describe three sets of methods that allow us (1) to model NPCs (finite state machines), (2) user behavior (agent based modeling of social network structures), and (3) bots (agent behaviors, automated planning techniques, consensus and coordination techniques). Our objective in future research is to study user behavior using social network analysis and natural language processing in order to establish agent based models of their behavior that will be mapped into an ontology of organizational design of large-scale MAS (developed partially in [7], [8], [9], [10]) to be used in the implementation of bots to foster organizational behavior among them.

MMORPGs have various subgenres, but a usual setting is that a protagonist is placed into a world in which he interacts with various NPCs and mobs which give out quests that have to be resolved in order to proceed to higher levels, buy better equipment or learn new skills like magic and similar. For the sake of the ModelMMORPG project the MMORPG The Mana World (TMW)[2] has been selected in order to conduct our research. It was selected due to a number of reasons: (a) it is open source (GPL licensed)) allowing us to modify code and add additional functionality, (b) it has a supportive community, (c) it supports a number of interaction techniques which can be studies (e.g. trade among players, IRC based chat, organizing teams called parties, social network functions e.g. friends, enemies, parties etc.), (d) it is a (more or less) finished game featuring lots of quests that can be analyzed.

[1]See http://ai.foi.hr/modelmmorpg for details.
[2]See http://themanaworld.org for details.

In order to study organizational behavior of players, apart from existing quests, we have partially developed a quest which fosters players to organize called "The quest for the dragon egg". The basic idea of the quest is that players have to establish parties of at least three players to transport a dragon egg to their castle to hatch it. Additionally, they have to gather a number of ingredients scattered across the Mana World to create a hatching potion. The catch is that time limits exist for each of the tasks, so they cannot be performed by a single player by himself. An egg hatches on a random location every 24 ± 1 hours, the egg has to be hatched in an hour or it goes bad. The hatching potion has to be available at the moment the egg is transported to the castle. Ingredients can be acquired by fighting mobs or by trading with other parties. Since all of these constraints are in fact (more or less) simple to implement, we use finite state machines to model NPCs needed for the quest.

The rest of the paper is organized as follows: in section II we give an overview of related work. In section III we show how NPCs can be modeled using finite state machines in TMW. In section IV we give an introduction on how to model social network behavior of players using an agent based technique. In section V we provide AI agent formalisms for modeling artificial players - bots. In the end VI we draw our conclusion and give an outline to future work.

II. Related Work

Since MMORPGs are a relatively new phenomena, there are consequently only few studies that investigate MAS and ABM approaches in MMORPGs. Most studies deal with the design of individual AI agents (NPCs). For example Eckschlager et al. [11] try to model emotional behavior through artificial neural networks and a Five-Factor Model of Personality. Tychsen & Hitchens [12] deal with the limited abilities of MMORPGs in story-telling and identify agent based story-telling [13] as a possible solution. Eladhari & Mateas [14] describe a technical framework for modeling personality and emotion for both players and NPCs called the Mind Module. They describe a semi-autonomous agent architecture built to be used in a multi-player environment, employed as a part of the player's avatar. Brom & Lukavsky [15] focus on the usage of full episodic memory in contrast to ad-hoc, or special purpose, memory, and how it can be contributed to the agent's believability.

Some investigations deal with a MAS oriented design of MMORPGs. Robert et al. [16] clarify why classifier systems (CS) are good candidates for action selection mechanisms of NPCs. CSs are used as control architectures for simulated animals or robots in order to decide what to do at each time. The authors describe different classifier systems and introduce a new CS architecture, acting in a MAS environment, which is adapted to the specific constraints of the MMORPGs. Fairclough & Cunningham [17] deal with the area of computer-mediated storytelling, and describe the "development of an expert case-based character director system which dynamically generates and controls a story, which is played out in a multi-player networked game world". The proposed system includes a story director system which utilizes the case based planning paradigm, and facilitates multi-player stories. Stories are modeled as cases, and their planning and scheduling is the primary activity of the system. This work is story-centered, and the notion of AI for the most part refers to the story director agent. Modeling of the NPCs is performed in a layered structure, from low-level behaviors to higher level targeted goals (low level such as collision detection, followed by social simulation, idle behaviors, targeted behaviors, attitudes etc.). Berger et al. [18] describe "an integrated, game-like e-Business environment that follows the role model of (...) MMORPGs", and essentially is a 3D virtual world based on game engine technology. This environment provides a platform for conducting business, and is based on MAS ideas, composed of many autonomous individuals - both human and software agents, which constitute avatars in the 3D virtual world. The goal of this environment is to support the complex interaction patterns between the members in an e-Business setting, and provide grounds for a lively and sustainable on-line community composed of providers and consumers.

Other studies employ ABM techniques to study player behavior. For example Lehdonvirta [19] is trying to apply concepts and techniques from economics to study the complicated interactions inside game worlds and is asking the question if techniques and theories from real economics be applied to virtual economics? Basic motivation for Lofgren & Fefferman [20] was a software update released into the MMORPG World of Warcraft, which created a full-blown epidemic. Players of this game encountered an extremely virulent and highly contagious virtual disease, which caused high rates of mortality and social chaos within gaming population that comes from a large-scale outbreak of a deadly disease. The authors conclude that the virtual outbreak in WoW game was both a missed opportunity to study epidemics, and a new direction for future epidemiological research. Ang & Zaphiris [21] have proposed a new method for sociability design and research through the use of simulation, and have developed an ABM to simulate and study the formation of social networks in a MMORPG guild community. Fishwick [22], on the other hand, introduces personal aspects of agents through first person perspective, inside an open-source, multi-user, virtual environment system called OpenSimulator.

While these studied do analyze the social structure and dynamics of players (trough social network analysis for example) they do not particularly target the question of how players organize in order to perform their tasks.

From this reasoning follows that there are two important aspects which aren't covered in contemporary MMORPG studies: (1) organizational techniques of artificial player's societies development, (2) study of organizational behavior of players.

III. Modeling NPCs

An NPC is an in-game character which is, most usually, used for communication with the player, serving the purpose of delivering all kinds of information to the player, e.g. basic information for new players, introduction to new areas, simple server-wide or local news service, information about a certain quest, etc. An NPC can range from a simple reactive agent to a completely interactive, environment and history immersed agent, based on the purpose it serves. Here we will consider simplest and an agent of average complexity, modeled as a finite automaton.

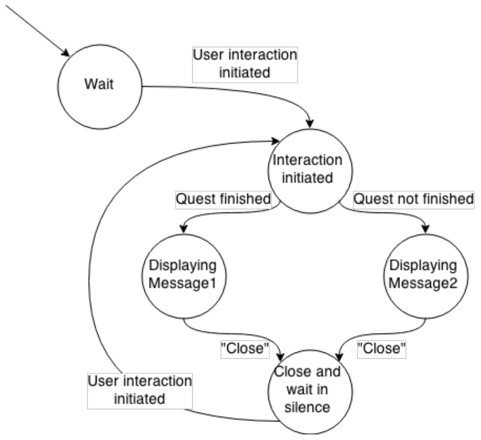
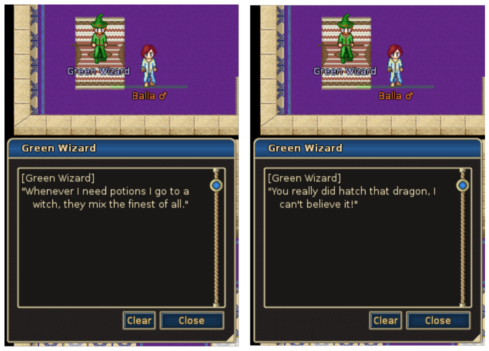


Fig. 1. Graph representation of the described agent



[Green Wizard]
"Whenever I need potions I go to a witch, they mix the finest of all."

[Green Wizard]
"You really did hatch that dragon, I can't believe it!"

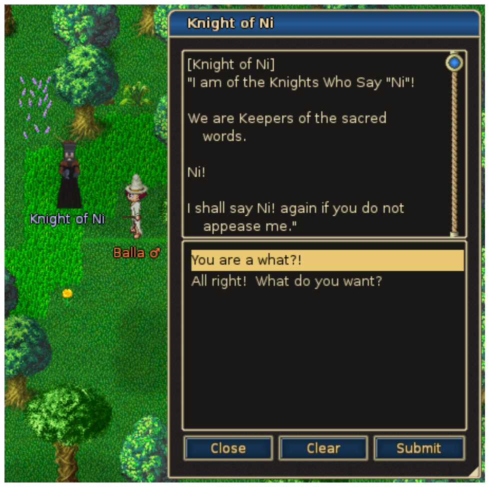
Fig. 2. In-game representation of the two simple interaction examples, dependent on the state of the quest

This finite state-machine needed for our goal is defined by six elements $(\Sigma, \Gamma, S, s_0, \delta, \omega)$, as follows: Σ is a non-empty symbol set called the in-alphabet, Γ is a non-empty symbol set called the out-alphabet, S is a finite non-empty set of states, s_0 is the initial state, δ is transition function defined as $\delta : S \times \Sigma \to S$, and ω is an exit function defined as $\omega : S \times \Sigma \to \Gamma$. This more general definition, when specialized for defining agents, is adapted as follows: in-alphabet is a set of possible experiences, out-alphabet is a set of possible actions, set of states is a set of agent's inner-self states, the initial state is the initial inner-self state, transition function is the next internal state function of the agent, whilst the exit is the action function that will be performed by the agent.

The less complicated version is a simple reactive agent which reacts to user-initiated interaction. Based on the status of the quest the user started with another NPC, the NPC delivers simple, static, information about the state of the world. As defined, the NPC has several defined states, activated by user interaction. Figure 1 shows an example of possible states and the transitions needed to reach them.

The in-alphabet of this agent consists of possible variable values, out-alphabet, considering it is a set of actions, gathers all the actions the NPC can take, i.d. send one of the messages to the player, set of inner NPC states are the states of this NPC, as visible in Figure 1, etc.

User interaction is started by clicking on the NPC or other



[Knight of Ni]
"I am of the Knights Who Say "Ni"!

We are Keepers of the sacred words.

Ni!

I shall say Ni! again if you do not appease me."

You are a what?!
All right! What do you want?

Fig. 3. In-game representation of the choices a player can make

ways of conversation initiation. The graph on Figure 1 depicts the possible state transitions, e.g. from *Interaction Initiated* state, in order to move to the *Displaying Message1* state, the quest status has to be checked. Based on this action, the agent changes its state, showing either *Message1* or *Message2*, as seen in Figure 2. Similarly, player has to click the Close button, so as to finish interaction with the NPC, and to get it ready for interaction with the next player, NPC or mob.

A more complex example is an NPC which gives out a quest, and is the sole authority on the state of the quest, depending on user interaction and a more dynamic environment, considering it interacts with the player on a more active level, and depends on some environment events, such as a timer. TMW NPCs are implemented using an domain specific scripting language, where quest process is tracked using a variable, and several code segments to be defined. These segments are accessed based on player decisions or status of variables or special elements of the environment. For example, an NPC, the Knight of Ni, after sending a default greetings message to the player, checks if this particular player is a part of a player party, and proceeds accordingly, as is shown in the code listing below.

```
if (getcharid(1)==0) goto L_Deny;
if (getcharid(1)) goto L_Allow;

L_Deny:
      (...)
L_Allow:
      (...)
```

The whole process of interaction between the user and the NPC is manipulated by using conditional expressions, such as in the following piece of code.

```
if (countitem(663) > 0 &&
     $@KnightNiShrubberyQuest == 3)
          set $@KnightNiShrubberyQuest, 4;
if ($@KnightNiShrubberyQuest == 0)
     goto L_Start;
if ($@KnightNiShrubberyQuest == 1)
     goto L_QuestState1;
```

In the code listing above, the first line checks if the player concerned has the needed item, defined by ID 663 (named *Black Rose* in the game), and if the quest state variable has value 3. If those are fulfilled, then the engine will set the value of variable *$@KnightNiShrubberyQuest* to 4. Other lines of code work in a similar manner, since the value of the quest state variable is checked, and a certain action is taken, e.g. the engine is instructed to go to the label *L_Start*, where some other code follows.

These mentioned labels are markers throughout the NPC script, which allow for non-linear script execution, and interactive communication with the user. For example, label *L_Start* contains the code which will try to attract the player to accept a quest. Although in a rather guided fashion, a player can choose the path they want to take, as shown in the code below.

```
L_Start:
    mes "We are
        Keepers of the sacred words.";
    next;
    mes "Ni!";
    next;
    mes "I shall say Ni! again if you do
        not appease me.\"";
    menu
        "You are a what?!",
            L_Ni,
        "All right! What do you want?",
            L_Describe;
```

Figure 3 depicts the example communication done in the game environment, at the moment when the player is offered to choose their move. Direct interaction with player data, such as player inventory content, is something this game genre cannot do without. The example NPC will proceed to direct player data interaction only if the quest state variable has a certain value, which is, in turn, set by checking the player's inventory. In case of a successful check, the NPC will proceed, taking an item from the player's inventory (*delitem*) and giving them another item (*getitem*). More specifically, when certain conditions for an action are met, the NPC is going to proceed to this defined action, hence producing an effect affecting the player, as shown in code listing below. This example represents a prime example of the finite state automaton, since there is an input, a condition for an action, the action and a certain effect this action has on the environment (i.e. the player).

```
L_Next4:
    mes "[Knight of Ni]";
    delitem 663, 1;
    mes "\"Very good,
         that is a nice shrubbery.";
    next;
    mes "You shall be awarded
        with this mighty gift!";
    getitem 742, 1;
    mes "Now go!\"";
    set $@KnightNiShrubberyQuest, 5;
    close;
```

Non-player interaction with NPC is enhanced by using NPC timers and special code which activates when this timer reaches a defined point in time. In this example, the NPC is going to change value of the quest variable exactly sixty minutes after the second part of the quest is started, as seen in code below.

```
L_Next2:
    (...)
    initnpctimer;
    close;

OnTimer3600:
    set $@KnightNiShrubberyQuest, 11;
    end;
```

This particular action allows for more unpredictable and non-linear NPC behavior, when observed by the player. Introducing a random number generating function would further increase unexpected, yet completely foreseen NPC behavior.

IV. Modeling Player Behavior

In order to model the behavior of players in an MMORPG one needs to acquire statistical data about them using the game. Such logging data in TMW is available in form of chat logs, user (social) behavior logs, user inventory logs, trade logs and other logs that can be collected and analyzed. The collected data can be analyzed using various methods, but two approaches are of particular interest: social network analysis (SNA) and natural language processing (NLP) with possible help of sentiment analysis (SA) [23], [24]. SNA allows us to study the social structure of the players (interactions including chat, trade, fight etc; friendships; antagonisms; parties and similar [25]) as well as processes on networks (the dynamics of network creation [26], spreading of information [27], mixing patterns [28], [29] etc.). While SNA is very insightful in the construction and analysis of player networks, the actual meaning (semantics) of the interactions as well as their intensity can more efficiently be analyzed using NLP and SA techniques. In this way a comprehensive study of behavior can emerge that can be the input to the construction of agent based models of player behavior (see [30] for detailed examples).

A very interesting method to be used here is the so called recipeWorld brought forward by Fontana & Terna which is "an agent-based model that simulates the emergence of networks out of a decentralized autonomous interaction" [31]. The main of their approach is that modeled agents are given so called recipes that represent variable numbers of steps, some of which might run in parallel, which ought to be taken in order to achieve a given result. In this way the social network of agents emerges as a side effect of agent behavior which use recipes to achieve attended objectives. The model of recipeWorld is founded on four distinct sets: A the actual world populated by entities and their actual network; B an ABM with agents that base their behavior on the objectives and recipes derived from A; C the social network generated by B, D the data exchanged on the network. Fontana & Terna proposed that it is possible to populate sets A, B and C, by knowing D with use of inference and an approach similar to reverse engineering.

From our perspective recipes can be patterns of interactions (e.g. message passing, trade transactions and other actions) that lead to well connected subnets over some observed periods

TABLE I. THE STRIPS ALGORITHM

If on top is	Then
A complex or atomic goal which is satisfied	Remove it
A complex goal which is not satisfied	1. Leave it on top 2. Put its subgoals on top in a different order
An atomic goal which is not satisfied	Find a rule which adding list includes the goal 1. Exchange the goal with the rule 2. Put the rule's preconditions on top If no such rule exists put the preconditions on top in a different order If no such order exists find an alternative rule. If no such rule exists the, goal is not satisfiable.
A rule	1. Remove the rule 2. Update the state using the rule 3. Write down the rule (solution)
Nothing	Stop

of time. These patterns of interaction could be extracted using adequate SNA and NLP techniques and then used for simulation experiments to construct artificial agent networks. Such simulated networks could then be analyzed and compared to empirical data to see if they actually conform to real processes by using various network metrics for example. In this way automated organization techniques for agents could be established.

V. MODELING BOTS

The previous contemplation brings us to the most complex part of the project which deals with implementation of artificial agent players. In order for a an artificial agent to play a MMORPG a number of MAS and AI techniques have to be employed. The agent has to have the ability to connect to the game server, learn about its environment, understand instructions from NPCs and mobs, solve quests and interact with other (possibly artificial) players and organize to solve harder (social) quests. For the first part, we have partially implemented a low level client to the TMW server in the Python scripting language that allows us to connect an agent to the server, move around, collect items and interact with other players, NPCs and mobs.

In order to implement intelligent agents we will use the Smart Python Agent Development Environment (SPADE) [32] that provides a FIPA (Foundation for Intelligent Physical Agents) compliant development environment for Python and features a number of predefined agent behaviors, advanced knowledge bases and agent organization facilities like group chat and organizational units. We will use SPADE to build an agent layer above the low level interface that will allow agents to understand their surroundings (by using some of the provided advanced knowledge bases), solve quests (by implementing a belief, desire, intention – BDI agent model and an automated planning algorithm e.g. STRIPS [33] see table I) and interact with their environment (using the integrated communication and organization facilities).

Individual agents are usually defined by a number of behaviors which might be [34]:

- **role factory** (a role added at runtime and then enacted by the agent);
- **itinerary** (allows mobile agents to travel across various locations and perform tasks);
- **periodic** (looped behavior possibly with a given period of time intervals between iterations);
- **observer** (an agents awaits an event in order to perform its actions);
- **listener** (a special type of observer in which an agent awaits a special message of some other agent);
- **client/server** (resembles the client-server model);
- **one-shot behavior or task** (represents a simple task or activity);
- **finite state machine** (resembles a finite state machine in which every node is an activity to be performed);
- **sequential behavior** (a sequence of other behaviors);
- **parallel** (various behaviors are run in parallel).

Most of these behaviors will have to be used to implement a working bot (except maybe for itinerary, since our agents will not have to travel across the network). Role factories come to use in organizational behavior of agents. For example when an agent joins a party he might be assigned a role to act as the carrier of some item (e.g. the dragon egg) and has to enact this role (e.g. has to keep line of sight with two other party members). The periodic behavior can be used in multiple different occasions for example to harvest a given resource mobs of a given type have to be fought. The agent will loop this behavior until it has enough of a given resource. An observer behavior is likewise important – an agent will observe its environment and in case of an event (e.g. an attack of some mob) react accordingly. The listener behavior is similar – if a bot receives a message from another player it might choose to react. Client/server behavior are especially important for trading – (server) agents will offer some items and wait for other (client) agents to pay for them in order to deliver the good. Task behaviors are ubiquitous – every action (moving, attacking, talking etc.) are simple tasks to be performed. FSM behaviors on the other hand might come in handy to deal with various states of quests – e.g. when the player solves one part of the quest (by for example bringing some item to some NPC) it moves to another state (e.g. finding some other item). Sequential behaviors are important for the planning system – all plans will be sequences of (simpler) actions. Parallel behaviors on the other hand are obvious – while solving some task an agent has to be alert about events in the environment, thus the task behavior will run in parallel with some kind of observing behavior.

VI. CONCLUSION

In this work in progress paper we have shown some important agent based modeling techniques that allow us to study MMORPGs with a special accent on techniques to be employed in the ModelMMORPG project in which the TMW MMORPG has been chosen for study. We have firstly shown how to implement NPCs using finite state machines and how they can be implemented in the scripting language of TMW. Later we have provided some recommendations on how to collect, analyze and model behavioral and social player data using SNA, NLP, SA and ABM techniques. Based on the proposed ABM technique to model the social and organizational behavior of players, we have contemplated that artificial agents might implement their consensus and coordination techniques in the same way by using the recipeWorld approach. In the end we have shown some important modeling techniques for implementing artificial players – bots – by providing important agent behaviors as well as examples of their usage.

Out future research is aimed towards implementing the (dragon egg) quest that will foster organizational behavior of players. Afterwards we will conduct a large scale study with the developed quest by collecting server logs that will be analyzed in the described way. The analysis will yield agent based models of organizational behavior among players that we will embed into an ontology of organizational design techniques for the development of large-scale MAS. In the end we shall implement artificial agent organizations and test their performance against human players on the TMW platform.

Acknowledgment

This work has been supported in full by the Croatian Science Foundation under the project number 8537.

References

[1] Wikipedia Contributors, "Role-playing video game — wikipedia, the free encyclopedia," 2015, [Online; accessed 9-February-2015]. [Online]. Available: http://en.wikipedia.org/w/index.php?title=Role-playing_video_game&oldid=642919862

[2] ——, "Massively multiplayer online game — wikipedia, the free encyclopedia," 2015, [Online; accessed 9-February-2015]. [Online]. Available: http://en.wikipedia.org/w/index.php?title=Massively_multiplayer_online_game&oldid=646153954

[3] ——, "Massively multiplayer online role-playing game — wikipedia, the free encyclopedia," 2015, [Online; accessed 9-February-2015]. [Online]. Available: http://en.wikipedia.org/w/index.php?title=Massively_multiplayer_online_role-playing_game&oldid=642752380

[4] ——. (2014) Comparison of massively multiplayer online role-playing games. [Online]. Available: http://en.wikipedia.org/wiki/Comparison_of_massively_multiplayer_online_role-playing_games

[5] Global Collect. (2014) The global mmo games market: Payments, intelligence and trends. [Online]. Available: http://www.globalcollect.com/the-global-mmo-games-market

[6] NewZoo. (2012) The global mmo market; sizing and seizing opportunities. [Online]. Available: http://www.newzoo.com/infographics/the-global-mmo-market-sizing-and-seizing-opportunities/

[7] M. Schatten, "Active graph rewriting rules for modeling multi-agent organizational dynamics," in *Proceedings of the IBC 2012, 1st International Internet & Business Conference*, M. Ivković, M. Pejić Bach, and V. Šimičević, Eds. Rovinj: BIT Society, 2012, pp. 180–185.

[8] ——, "Reorganization in multi-agent architectures: An active graph grammar approach," *Business Systems Research*, vol. 34, no. 1, pp. 14–20, 2013.

[9] M. Schatten, P. Grd, M. Konecki, and R. Kudelić, "Towards a formal conceptualization of organizational design techniques for large scale multi agent systems," *Procedia Technology*, vol. 15, pp. 577–586, 2014.

[10] M. Schatten, "Organizational architectures for large-scale multi-agent systems' development: An initial ontology," *Advances in Intelligent Systems and Computing*, vol. 290, pp. 261–268, 2014.

[11] M. Eckschlager, R. Bernhaupt, and M. Tscheligi, "Nemesys: neural emotion eliciting system," in *CHI'05 Extended Abstracts on Human Factors in Computing Systems*. ACM, 2005, pp. 1347–1350.

[12] A. Tychsen and M. Hitchens, "Ghost worlds–time and consequence in mmorpgs," in *Technologies for Interactive Digital Storytelling and Entertainment*. Springer, 2006, pp. 300–311.

[13] R. Aylett, S. Louchart, J. Dias, A. Paiva, M. Vala, S. Woods, and L. Hall, "Unscripted narrative for affectively driven characters," *Computer Graphics and Applications, IEEE*, vol. 26, no. 3, pp. 42–52, 2006.

[14] M. P. Eladhari and M. Mateas, "Semi-autonomous avatars in world of minds: A case study of ai-based game design," in *Proceedings of the 2008 International Conference on Advances in Computer Entertainment Technology*. ACM, 2008, pp. 201–208.

[15] C. Brom and J. Lukavsky, "Towards virtual characters with a full episodic memory ii: The episodic memory strikes back," in *Proc. empathic agents, AAMAS workshop*, 2009, pp. 1–9.

[16] G. Robert, P. Portier, and A. Guillot, "Classifier systems as' animat'architectures for action selection in mmorpg," in *3rd International Conference on Intelligent Games and Simulation, at London*, 2002.

[17] C. Fairclough and P. Cunningham, "A multiplayer case based story engine," Trinity College Dublin, Department of Computer Science, Tech. Rep., 2003.

[18] H. Berger, M. Dittenbach, D. Merkl, A. Bogdanovych, S. Simoff, and C. Sierra, "Playing the e-business game in 3d virtual worlds," in *Proceedings of the 18th Australia conference on Computer-Human Interaction: Design: Activities, Artefacts and Environments*. ACM, 2006, pp. 333–336.

[19] V. Lehdonvirta, "Virtual economics: applying economics to the study of game worlds," in *Proceedings of the 2005 Conference on Future Play (Future Play 2005), Lansing, MI*, 2005.

[20] E. T. Lofgren and N. H. Fefferman, "The untapped potential of virtual game worlds to shed light on real world epidemics," *The Lancet infectious diseases*, vol. 7, no. 9, pp. 625–629, 2007.

[21] C. S. Ang and P. Zaphiris, "Simulating social networks of online communities: simulation as a method for sociability design," in *Human-Computer Interaction–INTERACT 2009*. Springer, 2009, pp. 443–456.

[22] P. A. Fishwick, "An introduction to opensimulator and virtual environment agent-based m&s applications," in *Simulation conference (WSC), proceedings of the 2009 winter*. IEEE, 2009, pp. 177–183.

[23] B. Pang and L. Lee, "Opinion mining and sentiment analysis," *Foundations and trends in information retrieval*, vol. 2, no. 1-2, pp. 1–135, 2008.

[24] M. Schatten, "Opinion mining on news portal comments – towards a Croatian sentiment database," in *Seminar za metodologijo in informatiko*. Novo Mesto: Fakulteta za informacijske študije, 2012.

[25] S. Wasserman and K. Faust, *Social Network Analysis ; Methods and Applications*, ser. Structural analysis in the social sciences. Cambridge University Press, 1994.

[26] A. Barrat, M. Barthélemy, and A. Vespignani, *Dynamical Processes on Complex Networks*. Cambridge University Press, 2008.

[27] Y. Moreno, M. Nekovee, and A. F. Pacheco, "Dynamics of rumor spreading in complex networks," *Physical Review E*, vol. 69, no. 6, p. 066130, 2004.

[28] M. E. J. Newman, "Mixing patterns in networks," *Physical review. E, Statistical, nonlinear, and soft matter physics*, vol. 67, no. 2, pp. 1–13, 2003.

[29] R. Fabac, M. Schatten, and T. Đuričin, "Social network mixing patterns in mergers & acquisitions-a simulation experiment," *Business Systems Research*, vol. 2, no. 1, pp. 36–44, 2011.

[30] M. Schatten, J. Ševa, and B. Okreša Đurić, "An introduction to social semantic web mining & big data analytics for political attitudes and mentalities research," *European Quarterly of Political Attitudes and Mentalities EQPAM*, vol. 4, no. 1, pp. 40–62, 2015.

[31] M. Fontana and P. Terna, "From agent-based models to network analysis (and return): the policy-making perspective," in *SwarmFest*. University of Notre Dame, 2014.

[32] M. E. Gregori, J. P. Cámara, and G. A. Bada, "A jabber-based multi-agent system platform," in *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*. ACM, 2006, pp. 1282–1284.

[33] R. E. Fikes and N. J. Nilsson, "Strips: A new approach to the application of theorem proving to problem solving," *Artificial intelligence*, vol. 2, no. 3, pp. 189–208, 1972.

[34] T. Marian, B. Dumitriu, M. Dinsoreanu, and I. Salomie, "A framework of reusable structures for mobile agent development," in *Proceedings of IEEE International Conference on Intelligent Engineering Systems (INES2004)*, Cluj-Napoca, Romania, 2004, pp. 279–284.