Agent-Oriented Methodology for Designing 3D Animated Characters

Gary Loh Chee Wyai[1], Cheah WaiShiang[2] and Nurfauza Jali[2]
[1]*School of Computing, University College of Technology Sarawak, Sarawak, Malaysia.*
[2]*Faculty of Computer Science & Information Technology,*
Universiti Malaysia Sarawak, 94300 Kota Samarahan, Sarawak, Malaysia.
gary@ucts.edu.my

***Abstract*—Agent Oriented Methodology (AOM) has been used as an alternative tool to modelling the production of 3D animated characters. Besides allowing strong engagement between production team members, the agent models also drive effective communication among them. This paper explores the adoption of AOM to model the cognitive capability of 3D animated characters. We extend and demonstrate how AOM can be used to model a BDI (Belief/Desire/Intention) cognitive architecture for 3D animated characters in a fire fighting and evacuation scenario. The contribution of this work is that it turns the AOM into a detailed design tool for a 3D production team. Although the AOM can serve as an engagement tool among various stakeholders, we further showcase the use of AOM as a tool for production design and development.**

***Index Terms*—Agent-Oriented Software Engineering; BDI; Cognitive Architecture; Cognitive Modelling; Methodology.**

I. INTRODUCTION

Agent oriented software engineering (AOSE), or also known as Agent Oriented Methodology (AOM), is a software development practice that autonomously reduces the complexity of software development for dynamic systems [4]. It is especially useful for cooperative software that largely contains interactions [2] or software in an open and dynamic organizational environment [3]. For example, an electronic auction system requires software components to interact with each other in order to perform tasks. This can include making a decision task on behalf of users, negotiating a deal, deriving bidding strategies, immediately and proactively reacting to user requests, and identifying opportunities with or without human intervention.

Complex interactions that emulate human communications require reasoning capabilities that necessitates tools for decomposition, abstraction and organization. Conventional methodologies fail to support such practice since there is a gap in conceptual representation [3]. Hence, AOM are ushered in to bridge this gap [3]. According to [1], around 100 AOMs have been introduced. Some of the methodologies lack generality where they only focus on specific systems and agent architectures [5]. In addition, some of the methodologies have insufficient detail to have practical value [6].

AOM was initially introduced by Sterling and Taveter [5], at Melbourne University. It has been further extended to support rapid prototyping of socio-technical systems [5], information finding [26], e-commerce [7], sustainable software [9], video surveillance [13], environmental study[14] and collaborative games [8]. It can also be adopted as a "standard" agent methodology for industry [10] or novice developers who are engineering a complex system. AOM introduces a unified way to engineer a socio-technical system from analysis, design to implementation. Since current agent methodologies focus too much on specific domains, this unification is able to bridge the gaps among these methodologies.

When designing and implementing a socio-technical system, modelling activities are treated in series of stages. The modelling process consists of conceptual domain modelling, platform independent design and platform specific design. More specifically, the modelling process involves modelling the goals, roles, interactions and domain knowledge. This is followed by deciding on agent types, knowledge of agents, interactions between agents and agent behaviors. In addition, each model can be transformed into another model.

AOM has been explored as an alternative tool in the 3D animation character production industry [11] where the agent models are used to engage the production team. This paper continues the exploration of the adoption of AOM in this domain to model the cognitive capability of 3D animated characters. We extend and demonstrate how AOM can be used to model a BDI (Belief/Desire/Intention) cognitive architecture for the 3D characters in a fire fighting and evacuation scenario. The contribution of this work is to turn the AOM into a detailed design tool for the production team. Although the AOM can serve as an engagement tool among various stakeholders, we will attempt to showcase the usage of AOM as a tool for production design and development. This is important in order to align current production design and development practices with AOM.

This paper is organized as follows. Section II presents a review on cognitive modelling of 3D animation characters. The case study is elaborated in Section III, where the two scenarios of firefighting and evacuation is presented. Section IV outlines the combination details of the AOM and Prometheus agent-oriented software engineering methodologies for designing virtual characters with BDI architecture. The proposed methodologies combination covers the understanding of the problem domain for which the virtual characters are to be designed by conceptual domain modelling. Section V discusses on designing the BDI architecture for characters of the given problem domain by platform-independent design. Section VI addresses the implementation of the agent models created in Section IV in an object-oriented agent programming language (OOAPL). Finally, the conclusions and perspectives for future work are presented in Section VII.

II. Related Works

AOM was validated in the production process of 3D animated characters, where 12 undergraduate students from Swinburne University were selected as subjects [11]. The agent models were used to model the production process as a guide in the animation process. For example, an animator will animate a character with the goal of producing high quality and believability; a goal of creative activities and rigid activities. AOM allows the animator to evaluate his/her activities in a live production environment. In this case, the animators can simplify the communication and the expectations within the animation process. The agent models are able to engage and promote communication among production team members.

Based on the success study in [11], this paper presents the modelling of cognitive architecture for virtual character using agent models. BDI cognitive architectures have been used to model and control believable software agents [4]. From the reviews, most of the works are focused on integrating agent programming platform into games engines. For examples, work has been done to integrate BDI programming platforms like AgentSpeak, GOAL [19], Jason [21], 2APL [16], JACK, Jadex [20] into games engines like Open Wanderland [15], Unity [16] [18], Unreal engine [20].

Based on the mentioned works, there is neither a methodology nor a systematic process to model the detailed cognitive architecture of agents in serious games. As a result, it is hard to model, design and develop a cognitive agent among novice developers. Also, it is hard to transfer the same cognitive design to other similar projects. Hence, there is a need for a systematic process to model the cognitive agents in games.

A systematic methodology was introduced in [22] for cognitive modelling based on the natural complexity and variability of ordinary human behavior. This methodology provides (i) the notations to formulate the properties of cognitive mechanisms, and (ii) a way of executing or animating the theory of cognition to explicitly support the implementation details. In addition, it promotes a sharing of the same terminologies, annotations, models and development processes as stated in [23].

To fill the gap in designing autonomous and believable cognitive agents for 3D virtual worlds, AOM will be explored in this work to model a cognitive architecture for cognitive agents. The BDI cognitive architecture is adopted as it able to mimic human behavior and simple to implement.

III. Motivational Case Study

In this section, a motivational case study on cognitive agents participating in a scenario of firefighting and evacuation is presented. The scenario is used in Section 4 to validate the combined methodology for designing the multi-agent BDI cognitive architectures. The scenario is describe as following: VirtualAgent1 is in the building and VirtualAgent2 is in the open space. VirtualAgent1 has no fire extinguishing experience, while VirtualAgent2 is well trained to extinguish small fires. A fire suddenly burst out in the open space. VirtualAgent1, who is located in the enclosed space is unaware of the fire and continues with its work. VirtualAgent2, being located in the open space, will take action. Its first action is to find a fire extinguisher. Then, he will take the extinguisher, locate the fire, move towards the fire, and extinguish the fire.

In the following section, we present how to model the cognitive agent through AOM.

IV. Methodology for Designing Cognitive Agents

AOM is a comprehensive agent methodology that is developed through a viewpoint framework [5]. The viewpoint framework is a conceptual framework that has been introduced by Sterling and Taveter [5]. The framework introduces levels and aspects that are required to focus on when people are involved in engineering open distributed systems. The viewpoint framework is designed with a reduced number of aspects, as compared to the Zachman framework [5], to allow people to grasp the aspects more easily. In addition, the viewpoint framework is compliant with a model-driven architecture (MDA).

When designing and implementing a socio-technical system, a sequence of modelling activities is involved as shown in Figure 1. Briefly, the modelling process of AOM covers the abstraction layers of (i) conceptual domain modelling, (ii) platform-independent design, and (iii) platform-specific design and implementation. The conceptual domain modelling layer constitutes the system's high-level motivation layer. It describes the level that allows non-technical stakeholders of any given problem domain to elicit, represent, understand, and discuss the requirements for the designed system. The highest layer is not dedicated to any technology to be used for designing the system. The platform-independent design layer corresponds to the designer view of the system in which the design of the system is decided and represented. However, the design descriptions presented at this layer are not related to any particular implementation platform or language. The design layer instead, provides a description that can be converted into a particular implementation at the next layer – the platform-specific design and implementation layer. The design description at this layer allows the system to be deployed and executed in a particular environment of a specific platform, hardware configuration, technology, and architecture.

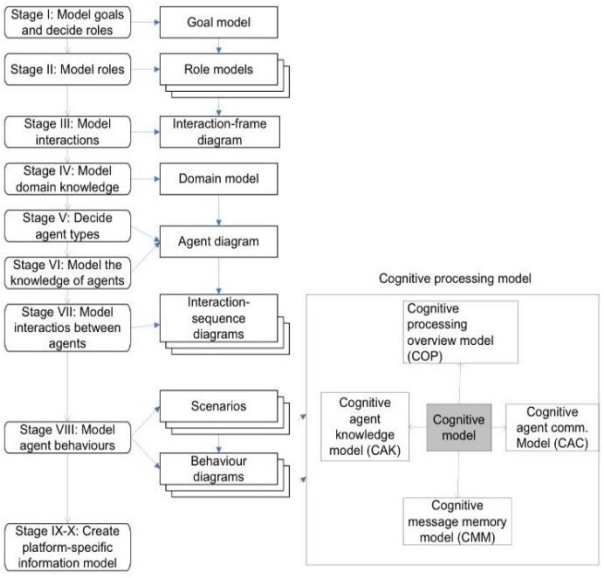

Figure 1: Extension of the AOM methodology by Prometheus for cognitive agents

From the Figure 1, it shows an extension of the stage VIII with the Prometheus models. The extension is needed to cater for concrete design of the multi-BDI cognitive architecture as described in the previous section. Due to the fact that AOM does not covers the BDI architecture, the integration of AOM with Prometheus [9] is needed in this research. As Prometheus is a methodology and modelling technique for systems of BDI agents, it is worth to adopt it in this research. Just like the work in [12], we claim that the AOM is able to support an effective requirement elicitation and analysis of cognitive processing and the Prometheus is able to support an effective initial cognitive design. Hence, a comprehensive methodology is introduced for cognitive agents.

In the cognitive processing modelling, we first model the details of the firefighting scenario. Thereafter, we represent the cognitive processing overview model and cognitive memory-message agent model for the scenario. Finally, the cognitive internal interaction model and cognitive knowledge model are formed to model the details of the cognitive configuration in relation to the given scenario. Cognitive processing modelling is an iterative process until the design goals are satisfied. The details of the models created during the course of the cognitive processing modelling are as follows:

A. Cognitive processing overview model (COP-model)

The Prometheus system overview model is adopted for this purpose to present an overview of the multi-agent BDI cognitive architecture. The model represents the agent types, interaction protocols, perceptions and actions involved in the cognitive processing;

B. Cognitive memory-message agent model (CMM-model)

The Prometheus agent overview model is adopted to present the overall message flow and the memory utilization strategy for the given agent during cognitive processing;

C. Cognitive agent communication model (CAC-model)

The AOM interaction diagram is adopted to present the interactions between the agents involved in the cognitive processing; and

D. Cognitive agent knowledge model (CAK-model):

The AOM behavior model is adopted to present the agent's beliefs and intentions during the cognitive processing.

A methodology for modelling and designing cognitive agent in 3D virtual worlds has been presented in this section. In order to validate and elaborate on the methodology, a walkthrough example of the motivational case study is described in the following section.

V. Designing the Cognitive Agents in the Fire Extinguishing Scenario

According to the methodology proposed in Section IV, modelling activities begin with conceptual domain modelling. Here, the problem domain is analyzed and requirements are elicited and represented in order to design the system. One of the main model types created at the stage of conceptual domain modelling is the goal model.

Figure 2 presents an overall goal model for the fire extinguisher. The goal model contains the following components: the goal, sub-goals, quality goals and roles. A goal signifies the functional requirement of the system, which can be decomposed into sub-goals. Quality goals are non-functional requirements and they set a specific standard to be achieved to improve the quality of the goal such as to ensure customer's satisfaction. Roles describe the capacity or the position to achieve the goal and quality goals. The main goal of the fire extinguisher is to 'handle fire'. The goal is achieved by two people, named, 'trainedEmployee' and 'untrainedEmployee'. There are two sub-goals to support the main goal. The sub-goals are 'put down the fire' and 'cry for help'. The trainedAgent is dependent on the untrainedAgent2 as shown in the organization model in Figure 3.

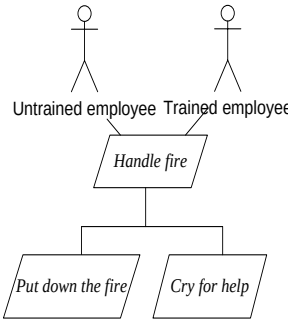

Figure 2: The overall goal model for the scenario of fire extinguishing

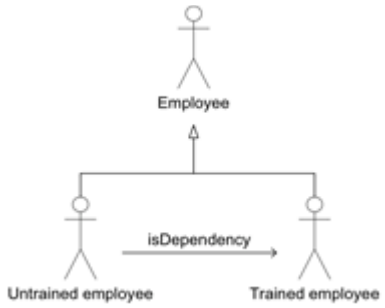

Figure 3: Organization model for the scenario of fire extinguishing

The domain model of the fire extinguishing scenario is shown in Figure 4. The domain model of AOM captures the knowledge to be represented within the designed system. Modelling domain knowledge involves identifying the domain entities and relationships between them. As illustrated in Figure 4, fourteen (14) domain entities have been modelled for the fire extinguishing scenario. Agents playing the role of Employee are situated in a building. The "Building layout" consists of "Physical objects" of types "Wall", "Fire", "Door", "Furniture", "Fire extinguisher", and "Window". All the physical objects are situated in the building and are modelled as contained by the "Memory" domain entity. Agents playing the roles of Trained Employee and Untrained Employee perform actions on the physical objects and perceive events associated with physical objects.

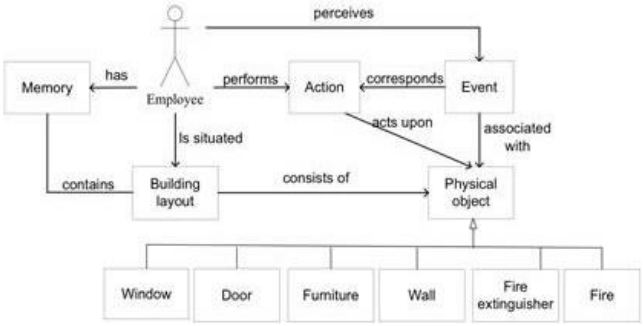

Figure 4: Domain model for the scenario of fire extinguishing

We present the higher-level model of fire extinguisher scenario through goal modelling, domain modelling. In the following section, we present the platform-independent design of the fire extinguisher scenario. As is shown in Figure 1, one of the central model types in platform-independent design are AOM scenario models. AOM Scenario models represent, for each scenario, the goal from the relevant goal model, the initiating agent, the triggering event, and the scenario description consisting of numbered steps of the scenario. Each step models one activity along with the condition for it to be performed, the role involved and agent type and the physical objects involved. Table 1 represents the high-level scenario for achieving the goal "Put out the fire". According to Table 1, the activity "Act on fire" is elaborated by another scenario – Scenario 2, which is represented as Table 2. The scenario modelled in Table 2 represents the cognitive processing within the multi-agent BDI cognitive architecture for the virtual character represented in Figure 1.

Table 1
The scenario for achieving the goal "Put down the fire"

Scenario 1				
Goal	Put out the fire			
Initiator	TrainedEmployeeAgent			
Trigger	Perceived event associated with the Fire object			
Description				
Condition	Step	Activity	Role / Agent type	Physical objects
	1	Act on fire (Scenario 1)	Trained Employee / Virtual Agent	Fire extinguisher Fire

Based on the scenario models of AOM, the next step is designing the cognitive capabilities of the involved agents by using models put forward by the Prometheus methodology. Figure 5 presents a cognitive processing overview model for the cognitive agents. This figure is modelled as a system overview diagram of Prometheus. The diagram represents the two agents involved in the fire extinguishing scenario. Different agents interact by means of the following protocols: "act on fire protocol", "cry for help protocol", "evacuation protocol". The "act on fire protocol" consists of simple rules that "notify" the other virtual agents about the effort to put out the fire. The "cry for help protocol" consists of rules to coordinate the fire extinguishing process among the virtual agents. Finally, the "evacuation protocol" consists of rules to coordinate the evacuation process among the virtual agents.

Both agents receive incoming perceptions of time, events, physical objects, and incoming communication and actions by other agents, from the environment. Actions are executed by sending the "execute action" commands to the body of the virtual agent.

Table 2
The elaborated scenario for achieving the goal "Put down the fire" by a virtual character

Scenario 1				
Goal	Act on fire			
Initiator	TrainedEmployeeAgent			
Trigger	Perceived event associated with the Fire object			
Description				
Cond.	Step	Activity	Agent types and roles involved	Physical objects
	1	Cognition configuration		
	1.1	Subscribe to the Fire object	Virtual agent	Fire
	1.2	Set attentions/state to idle mode		
	1.3	Subscribe to domain objects		Building layout
	1.4	Subscribe to domain objects		Building layout
	1.5	Subscribe to body locality		Locality
	2	Cognition to act on fire		
	2.1	Notify fire		Fire
	2.2	Update attention/state		
	2.3	adoptGoal(act on fire)		
	2.4	Deliberation and execute plan		
	2.4.1	Locate fire extinguisher		Fire Fire extinguisher
	2.4.2	Execute traversing plan		Building layout
Loop	2.4.3	Wait for action status		
	2.4.4	Grasp the fire extinguisher		Fire extinguisher
	2.4.5	Locate fire		Fire Fire extinguisher
	2.4.6	Execute traversing plan		Fire extinguisher
Loop	2.4.7	Wait for action status		Fire Fire extinguisher
	2.4.8	Put out the fire		Fire Fire extinguisher
	2.5	Update attention/state		

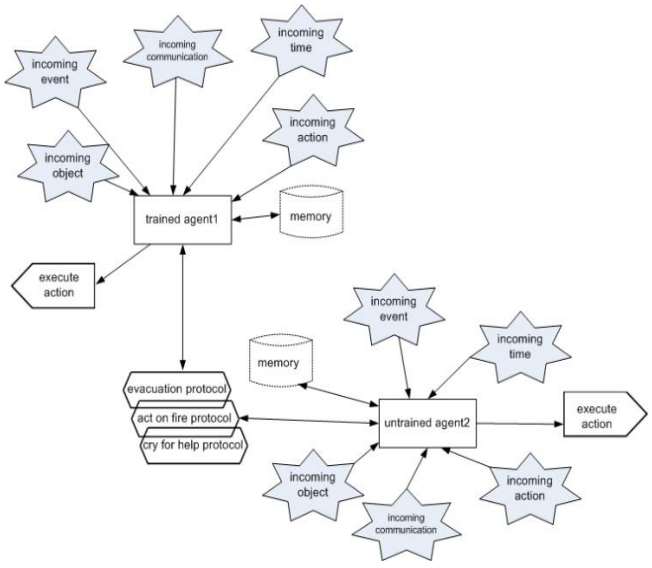

Figure 5: System overview diagram for a cognitive agent

Figure 6 represents a Prometheus agent overview diagram for the trained agent. This diagram shows the model for the

capabilities of an agent, the triggers of the capabilities and the execution of the capabilities. The trained agent has the following six capabilities: “ask for help”, “offer for help”, “update agent state”, “act on fire”, “grasp things”, and “use things”. Each capability modelled in Figure 7 receives inputs as messages or from the agent memory. The same model also represents strategies that support certain complex capabilities. For example, the “ask for help strategy” supports the “ask for help” capability, and the “explore strategy” and “traversing strategy” support the “act on fire” capability and finally the “offer help strategy” and “traversing strategy” support the “offer help” capability. Meanwhile, strategies also generate messages that request the execution of certain actions. In addition, a strategy may lead to another strategy or capability. For example, the “explore strategy” leads to the “traversing strategy” as well as to the “grasp things” and “use things” capabilities.

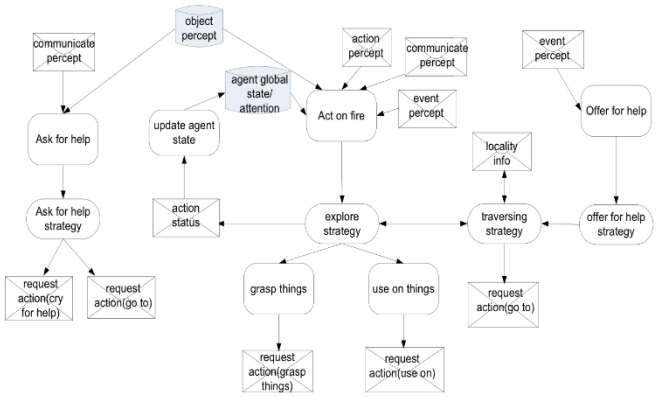

Figure 6: Agent overview diagram for the trained agent

In the first scenario, both agents do not communicate with each other. Figure 7 presents the interaction protocol for the second scenario. In this case, the untrained agent is “crying for help” by sending a communication message to the trained agent. The trained agent receives the message and responds with “agree to offer help”.

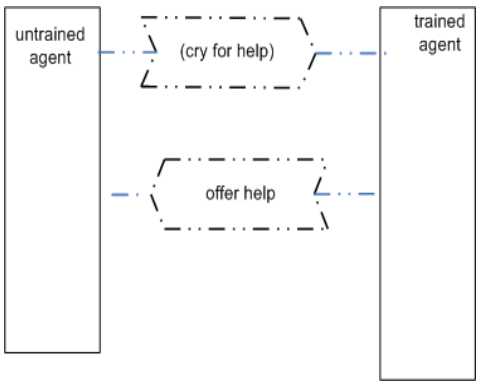

Figure 7: Interaction protocol for the second scenario

Figure 8 presents an overview of a behavior model for the trained agent. It models a deliberation of an agent towards the entire goal. It models the agent’s belief within a certain context, and the agent’s intention to achieve its desire. Also, it models belief update, intention reconsideration (e.g. Rule) and intention execution.

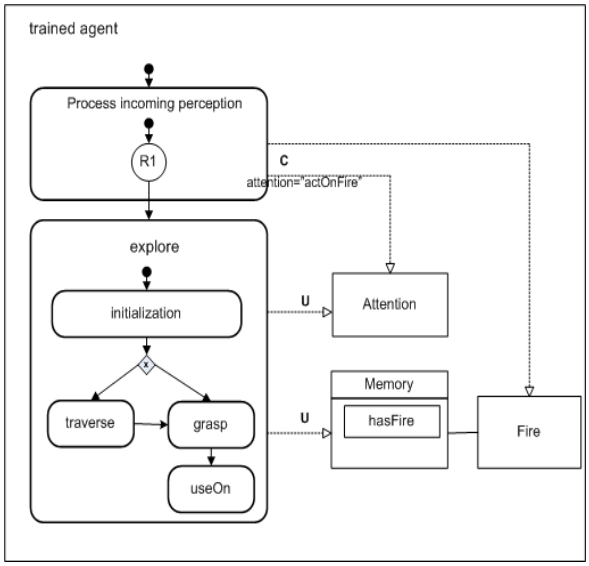

Figure 8: Behavior model for trained agent

A behavior model indicates what individual agents of a particular type do [10]. It enables both the modelling of proactive and reactive behaviors. An agent achieves a goal through performing activities, strategies or a plan. A rule is the basic behavior modelling construct. A rule is triggered due to an activity start event, conditions that have been fulfilled, or an action event caused by external agents. The execution of a particular activity is modelled by triggering a rule to update the agent’s mental state and/or send the message or perform an action of another type by an individual agent.

In Figure 8, the Trained Agent starts to deliberate during receiving of perception of fire object. The BDI interpreter updates the agent attention and creates a fire object in the agent belief. Hence, the Fire object and fire attention will trigger the deliberation of exploring strategy and plan. The exploring strategy consists of a sequential plan performing strategy initialization, triggering the traverse strategy or performing grasp action and finally performing the use on action. Meanwhile, the execution of the explore strategy leads to belief update.

VI. Implementation of the Fire Extinguishing Scenario

The previous section explained the platform-independent models for the scenario of fire extinguishing. In this section, we focus on the platform-specific design and implementation of the scenario in the object-oriented agent programming language (OOAPL) [25], based on the platform-independent models. OOAPL is a Java-based language that allows flexible control and scalability of the agent deliberation lifecycle for programming BDI agents. In brief, the agent deliberation lifecycle in OOAPL is parallel, concurrent and distributed. This guarantees a balance between slow and fast deliberation processes. The readers can be referred to [25] for a better understanding of OOAPL.

While the mind of a virtual character is implemented as an OOAPL agent, the body of the agent is implemented by the CIGA middleware. The CIGA middleware [16] supports the development of non-player characters in virtual worlds.

Figure 9 shows two screenshots of “putting out the fire” by the virtual characters. During the simulation, the

virtualAgent1 is situated at an open space and the virtualAgent2 is situated in a room. The virtualAgent1 has fire extinguishing knowledge whereas the virtualAgent2 does not know how to put out the fire. Once a fire occurs in the room, the virtualAgent2 will cry for help. Then, the virtualAgent1 find the scream reactively. This is followed by asking the location of the virtualAgent2 by the virtualAgent1. Then, the virtualAgent1 locates the fire extinguisher, moves to the fire extinguisher, grasps the fire extinguisher, moves towards the location of the virtualAgent2, locates the fire and uses the fire extinguisher to extinguish the fire.

Figure 9: Virtual agents dealing with the fire

VII. CONCLUSION

Cognitive processing within a believable virtual character is complex. This complexity can be tamed by using multi-agent technology in building such virtual characters, which results in the Agent Oriented Methodology. This paper presents the combination of the AOM and Prometheus methodologies for modelling the cognitive capabilities of agents in 3D virtual worlds. An extension of AOM by Prometheus is required because AOM does not support the design of BDI agents. Through this combined methodology, cognition by a software agent for a virtual character is modelled at the abstraction layers of conceptual domain modelling, platform-independent design, and platform-specific design and implementation. In summary, the proposed combined methodology supports conceptualization of cognitive agents, which is closer to the concerns of the problem domain at hand and is easier to understand and validate. It also improves the efficiency and quality of the cognitive agent development process. Furthermore, the proposed methodology reduces the complexity of developing cognitive agents. In the future, more empirical studies are required to further justify the benefits of the combination of AOM and Prometheus in designing cognitive agents for virtual characters.

REFERENCES

[1] P. Koutsabasis, and J. Darzentas, "Methodologies for agent systems development: Underlying assumptions and implications for design," *AI & Society*, vol. 23, no. 3, pp. 379-407, 2007.

[2] C. Bernon, M. Cossentino, and J. Pavón, "Agent-oriented software engineering," *The Knowledge Engineering Review*, vol. 20, no. 2, pp. 99-116, 2005.

[3] L. M. Cysneiros, V. Werneck, J. Amaral, and E. Yu, "Agent/goal orientation versus object orientation for requirements engineering: A practical evaluation using an exemplar," in *Proc. of VIII Workshop in Requirements Engineering*, 2005, pp. 123-134.

[4] V. Silva, A. Garcia, A. Brandão, C. Chavez, C. Lucena, and P. Alencar, "Taming agents and objects in software engineering," in *SELMAS 2002: Software Engineering for Large-Scale Multi-Agent Systems*, 2003, pp. 1-26.

[5] K. Taveter, and L. Sterling, *The Art of Agent-Oriented Modelling*. MIT Press, 2009.

[6] A. Sturm, and O. Shehory, "Agent-oriented software engineering: Revisiting the state of the art," in *Agent-Oriented Software Engineering*, O. Shehory and A. Sturm, Eds. Berlin, Heidelberg: Springer, 2014, pp. 13-26.

[7] C. WaiShiang, A. B. Masli, and E. Mit, "Sustainability modelling of e-Commerce for rural community: A case from Long Lamai e-Commerce initiative," in *Proc. of IEEE International Conference on Informatics and Creative Multimedia (ICICM)*, 2013, pp. 282-287.

[8] C. W. Loh, C. WaiShiang, A. K. Chowdhury, and C. Gulden, "Engineering sustainable software: A case study from offline computer support collaborative annotation system," in *Proc. of 9th IEEE Malaysian Software Engineering Conference (MySEC)*, 2015, pp. 272-277.

[9] C. WaiShiang, E. Mit, and A. A. Halin, "Shared single display application: An interactive patterns approach," *Journal of Software Engineering and Its Applications*, vol. 9, no. 2, pp. 233-250, 2015.

[10] T. Miller, B. Lu, L. Sterling, G. Beydoun, and K. Taveter, "Requirements elicitation and specification using the agent paradigm: The case study of an aircraft turnaround simulator," *IEEE Transactions on Software Engineering*, vol. 40, no. 10, pp. 1007-1024, 2014.

[11] S. Murdoch, "Agent-oriented modelling in the production of 3D character animation," *Studies in Australasian Cinema*, vol. 10, no. 1, pp. 35-52, 2016.

[12] A. A. Letichevsky, "Theory of interaction, insertion modeling, and cognitive architectures," *Biologically Inspired Cognitive Architectures*, vol. 8, pp. 19-32, 2014.

[13] C. WaiShiang, O. B. Tien, T. F. Swee, M. A. Khairuddin, and M. Mahunnah, "Developing agent-oriented video surveillance system through agent-oriented methodology (AOM)," *Journal of Computing and Information Technology*, vol. 24, no. 4, pp. 349-368, 2016.

[14] J. V. Berna-Martinez, and F. Marcia-Perez, "Robotic control systems based on bioinspired multi-agent systems," *International Journal of Advanced Engineering Sciences and Technologies*, vol. 8, no. 1, pp. 32-38, 2011.

[15] C. WaiShiang, S. YeeWai, S. Nizam, and C. W. Loh, "Agent oriented requirement engineering for lake mathematical modelling: Preliminary study," *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, vol. 8, no. 2, pp. 5-10, 2016.

[16] O. Van, W. Joost, and D. Frank, "Goal-based communication using BDI agents as virtual humans in training: an ontology driven dialogue system," in *Proc. of Agents for games and simulations II*, 2011, pp. 38-52.

[17] Y. Luo, L. Sterling, and T. Kuldar, "Modelling a smart music player with a hybrid agent-oriented methodology," in *Proc. of 15th Requirements Engineering Conference*, 2007, pp. 281-286.

[18] P. R. Smart, T. Scutt, K. Sycara, and N. R. Shadbolt, "Integrating ACT-R cognitive models with the Unity game engine," in *Integrating Cognitive Architectures into Virtual Character Design*, IGI Global, 2014.

[19] K. V. Hindriks, V. R. Birna, B. Tristan, K. Rien, K. Nick, P. Wouter, and D. R. Lennard, "Unreal goal bots," in *Proc. of Agents for games and simulations II*, Berlin Heidelberg: Springer, 2011, pp. 1-18.

[20] S. Korecko, S. Branislav and C. Pavol, "Emotional agents as non-playable characters in games: Experience with Jadex and JBdiEmo," in *Proc. of IEEE 15th International Symposium Computational Intelligence and Informatics (CINTI)*, 2014, pp. 471-476.

[21] C. Sioutis, I. Nikhil, and C. J. Lakhmi, "A framework for interfacing BDI agents to a real-time simulated environment," in *Design and application of hybrid intelligent systems*, 2003, pp. 743-748.

[22] R. Cooper, F. John, and S. Tim, "A systematic methodology for cognitive modelling," *Artificial Intelligence*, vol. 85, no. 1-2, pp. 3-44, 1996.

[23] J. M. Gascueña, and F. Antonio, "Agent-based modeling of a mobile robot to detect and follow humans," in *Proc. of KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications*, Berlin Heidelberg: Springer, 2009, pp. 80-89.

[24] L. Padgham, and W. Michael, "Prometheus: A methodology for developing intelligent agents," in *Proc. of International Workshop on Agent-Oriented Software Engineering*, Berlin Heidelberg: Springer, 2002, pp. 174-185.

[25] M. Dastani, and T. Bas, "From multi-agent programming to object oriented design patterns," in *Proc. of International Workshop on Engineering Multi-Agent Systems*, 2014, pp. 204-226.

[26] C. WaiShiang, L. Sterling, and K. Taverter, "Task knowledge patterns reuse in multi-agent system development," in *Proc. of 13th International Conference on Principles and Practice of Multi-Agent Systems*, Kolkata, India. 2010, pp. 459-474.