AGENT-BASED COMPUTER-GENERATED-FORCES' BEHAVIOUR IMPROVEMENT

Mike Bourassa, Nacer Abdellaoui
Defence R&D Canada Ottawa, 2701 Carling Avenue, Ottawa, (K1A-0Z4), Ontario, Canada

Glen Parkinson
Xplornet, 2835 Ashton Station Road, Ottawa, (K0A-1B0), Ontario, Canada

Keywords: Artificial intelligence, Agent-based, Computer generated forces, Computer generated actors, Maslow's hierarchy of needs, JADE, Synthetic environment, Modeling and simulation.

Abstract: This paper captures the initial stages of a research project into improving the decision making performance of simulated entities in Computer Generated Forces (CGF) software applications. To date, the decisions made by Artificial Intelligence (AI)-enhanced synthetic entities have demonstrated a limited ability to react to changes in the synthetic environment, to use sensor data as effectively as a human operator, or in general to impact the synthetic environment in a comparable manner to a human operator. This paper presents a survey of AI in both the video gaming industry and academic circles leading to the proposal of a new agent architecture that combines a traditional agent architecture with a psychological framework (Maslow's Hierarchy of Needs) leading to the specification of a "Needs-based" agent. This paper also captures the initial design decisions on the construction of a prototype and identifies candidate technologies to advance the research to the next phase. It is proposed that by combining the cognitive elements of the psychological framework with the behavioural emphasis of agents, synthetic entities in military and non-military simulations can produce better decisions and therefore exhibit more realistic behaviour which by ricochet will require less human intervention in simulation executions.

1 INTRODUCTION

Simulation is a promising technology to prepare for a world of uncertainty, to acquire skill, or study alternatives. The Computer Generated Forces (CGFs) systems are the cornerstone of constructive simulations and an efficient way of providing extra players in a synthetic environment containing human participants. They are a viable alternative in experimentation, concept analysis and development, tactics development, and training. Existing CGFs are adequately designed for the symmetric mindset and well adopted to the Cold War era where all forces act according to Standard Operating Procedures (SOPs). With the fourth generation, non-kinetic warfare, and asymmetric warfare; SOPs are hardly ever followed to the letter, making current CGF systems increasingly inadequate. One of the key drawbacks of existing CGF systems is the lack of adequate representation of human influences such as perception, reasoning, decision making, or what is recognized as lack of Artificial Intelligence (AI). A comparative analysis about AI capabilities in CGFs concluded that these capabilities are very limited and recommended the realisation of a complementary AI component that should operate with existing CGFs. Also, the literature search that examined AI approaches for the video game industry and academia (Bourassa and Massey, 2009) and the AI in CGFs preliminary analysis (Taylor et *al*., 2009) identified software Agents as a promising technology to improve CGF entities' AI.

The goal of this paper is to evaluate the software agent system as a potential platform for enhancing the AI capability of CGF systems and to propose an agent based system to adequately represent human influences within CGFs, leveraging from academia research and video gaming technology.

Bourassa M., Abdellaoui N. and Parkinson G..
AGENT-BASED COMPUTER-GENERATED-FORCES' BEHAVIOUR IMPROVEMENT.
DOI: 10.5220/0003187002730280
In *Proceedings of the 3rd International Conference on Agents and Artificial Intelligence* (ICAART-2011), pages 273-280
ISBN: 978-989-8425-41-6

2 ARTIFICIAL INTELLIGENCE IN COMPUTER GENERATED FORCES

CGF entities are known in the gaming industry and academia as the Non-Player Characters (NPC) or Computer Generated Actors (CGA). While a certain amount of AI is built into most CGFs, it has historically been quite rudimentary, leading to the need for human intervention to achieve realistic behaviour. The purpose of this project is to find a methodology to improve the AI capability of any CGF and thereby reduce the level of human involvement in simulation executions. The goal of defining the artificial intelligence (AI) approach for the control of CGF entities began with a literature search. This examined AI approaches for both the video game industry and academia; which led to the exploration of agent-based systems and cognitive architectures, and the selection of agent-based systems for future consideration. Based on this decision, types of agent architectures were reviewed and found wanting, leading to the development of a hybrid agent concept based on “needs”.

2.1 Literature Search

The conclusions of the literature search (Bourassa and Massey, 2009) were surprising. The video gaming industry at the time of writing did not prove to be a good source of guidance for AI that exhibits realistic behaviour. The focus in the gaming industry is on a positive player experience not necessarily a realistic one. A positive experience is created by devoting a significant share of computer resources to impressive graphics and ensuring challenging but winnable games. AI is relegated a very small percentage of processing time. In fact, true AI implementations in games are rare because it is possible to make CGAs appear intelligent by allowing fast reaction times, providing clever scripting, and giving CGAs access to information the player does not have. As a result, academia remains the principal forum in AI development. The literature search traced the progress of academic AI research from the early attempts at symbolic logic solutions through machine learning approaches and finally to agent paradigms. All of the approaches have had some success but none have been adopted as general solutions to implementing AI in games. The agent paradigm had considerable success in many fields and is an intuitive way to conceive CGAs; therefore, it was decided to use an agent approach in this work.

2.2 Cognition

In order to implement a mapping of perceptions to actions (i.e. behaviour), we chose to use an "agent" paradigm. The agent is a combination of architecture and software that encapsulates autonomy (mapping perceptions to actions) and communication. The architecture defines the interaction with the chosen environment, while the software defines the nature of that interaction. An agent does not in and of itself address AI. Software agents are used in network centric architectures without any AI or general intelligence. Since cognition must be built into an agent, it is important to understand the nature of cognitive architectures to appreciate the limitations inherent in existing agent paradigms. This section addresses historic approaches to cognition and agents.

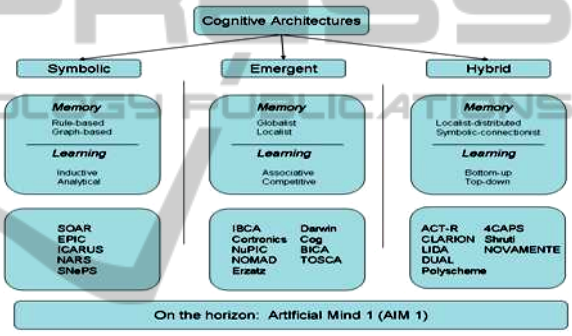

Figure 1: Simplified Taxonomy of Cognitive Architectures.

Duch (Duch et *al*., 2008) outlined a taxonomy of cognitive architectures (Figure 1). The main headers (symbolic, emergent, hybrid) reflect the evolution of cognitive architectures. The symbolic group typified by State, Operator and Results (SOAR) is very much a first order logic approach to cognition. The emergent group use approaches inspired by connectionist principles seeking to replicate brain functionality by modeling brain components (e.g. CORTRONICS models the biological functions of the thalamocortex in the human brain) or mimicking them through machine learning techniques (e.g. neural networks). Finally, the hybrid approaches try to combine symbolic and emergent techniques hoping to build on the strengths of the two. Adaptive Components of Thought – Rational (ACT-R) (Anderson, 1993) seems to be the most well-known of the current hybrid techniques.

The literature search found none of the preceding cognitive architectures has been successfully integrated into games. The principal problem is

likely that these architectures are meant to represent a single entity and cannot be scaled up to a multi-entity game. To be represented in a general and suitable manner; simpler approaches must be considered and therefore our focus on agents.

2.3 Agents and Behaviour

A CGA agent is intended to exhibit behaviour that approximates that which a human player would expect from another human player. In the CGF context, a human player should not be able to discern whether his opponent is a non-player character (NPC) or another human. There are several agent architectures available for consideration (Russell and Norvig, 2003). These are simple reflex, model-based and goal/utility-based agent architectures. Each of these agent architectures receives information about the environment through "Sensors" and alters the environment with "Actuators". The difference between the various agent architectures is in the internal processing of the agent.

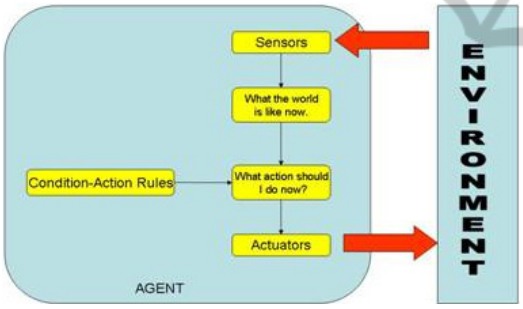

Figure 2: Simple reflex Agent Architecture.

Figure 2 illustrates the Simple Reflex Agent architecture. It receives information through the sensors and assesses the state of the environment based solely on the immediate sensor data. This architecture has no memory. Based on the immediate condition of the environment as measured by the sensors, it selects the appropriate actions and triggers the necessary actuators. These condition-action rules are usually implemented in the form of "if..then.." statements, decision trees, or a table of actions. Recent developments do however allow for the dynamic construction of decision trees to plan appropriate actions.

This type of architecture is moderately successful for low-level behaviours and is the standard for many current (and most past) agent implementations in the video game industry. The problem with this architecture is that the size of the condition-action table is a function of the number of the variables. Similar to symbolic cognitive architectures, this means that one must define all possible states and outcomes. This makes the simple reflex agent realizable only when the environment is specified by a limited number of variables. Finally, note that the only world model contained in the agent is the one implicit in the rules. That is, the rules are based on a world model known to the creator of the rules. The agent cannot create new rules or modify existing ones.

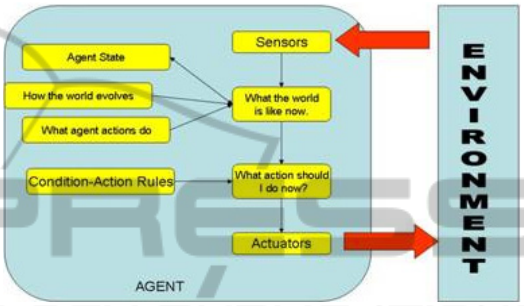

Figure 3: Model-based Reflex Agent.

The Model-based Reflex Agent architecture (Figure 3) tries to form a more complete picture of the world rather than just the raw sensor inputs. Model-based agents incorporate a sense of how the world evolves. In considering any future action, this architecture considers both the present state of the agent and the possible impact of any action on the world. The Model-based architecture comes closer to the ideal of autonomy (and to cognitive architectures) than the simple reflex agent by having this 'understanding' of the state of the world and the ability to track changes to this understanding. Even in the case of a partially observable world, Model based agents may be able to infer state information not directly observable. In this sense, such agents can recall the past and infer the future state of the environment. This allows for a process of learning, which is a prerequisite to achieving autonomy (Russell and Norvig, 2010). The challenges in creating such agents are in the implementation of capturing the dynamics of the world, determining the influence of actions on the world state, and establishing the mechanism and nature of agent learning. Any agent that depends on rules potentially faces the challenges of scalability seen with simple reflex agents.

Goal and Utility-based architectures (Figure 4 and Figure 5) extend the paradigm of the Model based architecture. Both have some knowledge of the world and how it evolves; however, the

difference is that Goal-based agents effect actions to achieve a particular goal and the Utility-based agents seek actions that satisfy a utility function. Anthropomorphizing the two types of agent, the Goal-based agent might ask 'Did I accomplish the mission?' while the Utility-based agent might ask, 'Am I happier with the world due to my actions?'.

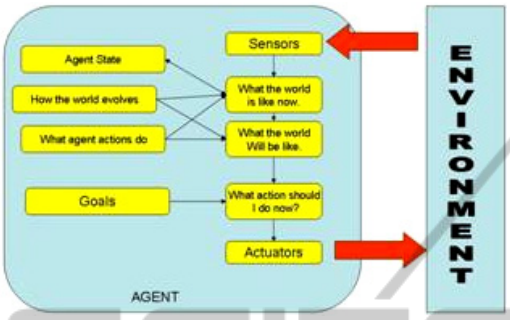

Figure 4: Model-based Goal-based Agent.

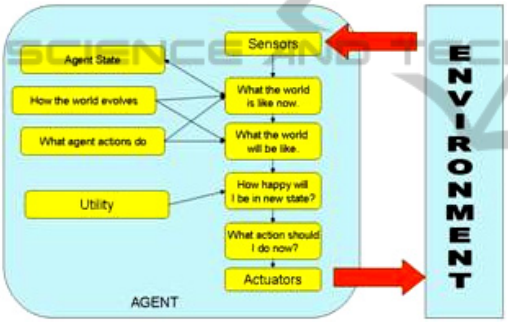

Figure 5: Model-based Utility-based Agent.

The agent types discussed in this section appear simpler than the cognitive architectures outlined earlier which is not surprising. The cognitive architectures seek to mimic thinking while the agents seek to mimic behaviour. In this regard, the agent architectures fall short of being able to provide completely autonomous behaviour in that they are still reflexive to their environment i.e. there is no reasoning. Furthermore, there is no adaptability as the architectures do not explicitly signal where, when, nor how learning should proceed. Is it possible then to improve agent behaviour without adopting the overhead of the existing cognitive architectures? The following proposes an agent architecture that attempts to align agent behaviour more closely with that of a human player.

2.4 Needs-based Agent

Two assumptions form the basis for the proposed agent architecture. First, it is a given that the human condition does not consist simply of reacting to the environment in terms of set goals or utilities. Even in the context of a game, a human player's actions may only be broadly described in terms of goals and utilities. In reality the human player's actions are modulated by other factors such as emotion or motivation (Ness, Tepe, and Ritzer, 2004). Thus, the agent architecture must include to some degree factors that modulate human behaviour. Secondly, it is a hallmark of all life forms that they be adaptable (Grand, 2000). Life is a continual learning process with survival as the goal. Therefore, the agent architecture must include mechanisms for learning and that implies cues for when learning must occur.

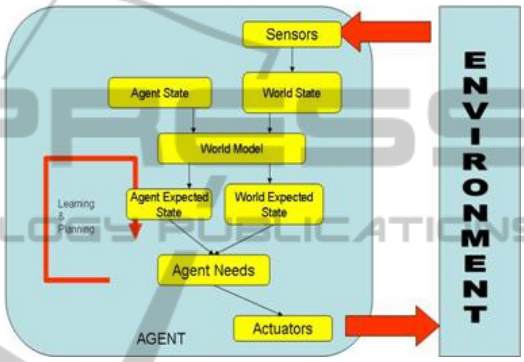

Figure 6: Proposed Agent Architecture.

Figure 6 is a sketch of a proposed Needs-based agent. The World State is the agent's representation of the world including its own state in the world. The Agent State represents the agent's internal state that is not necessarily visible to the world. The World Model describes how an agent's World State and Agent State will change in that world. The Expected State, for both the agent and the world, is the output of the world model and represents an estimate of how things may change in each case. The Agent Needs are a set of functions that reflect Maslow's Hierarchy of Needs (see Figure 7). Needs are tuplets of goals and utility as complimentary, simultaneous considerations.

The model is similar in structure to the model-based architecture but differs significantly in three respects. First, it incorporates the concept that the human mind formulates models of the world and continuously tries to predict its environment. This was expressed most recently by Hawkins (Hawkins and Blakeslee, 2004), more generally (and much earlier) by Brentano (Brentano and al, 1874), and to some extent by Grand (Grand, 2007). When predictions meet realization, the process is auto-associative i.e. the predicted outputs are identical to the next experienced inputs.

Second, learning is cued to occur when predictions conflict with reality to a significant degree (though the threshold must somehow be defined). The expected world and agent states are compared to the present input states. If the world model is satisfactory there will be no difference between the two states; however, any difference in the two will potentially trigger learning.

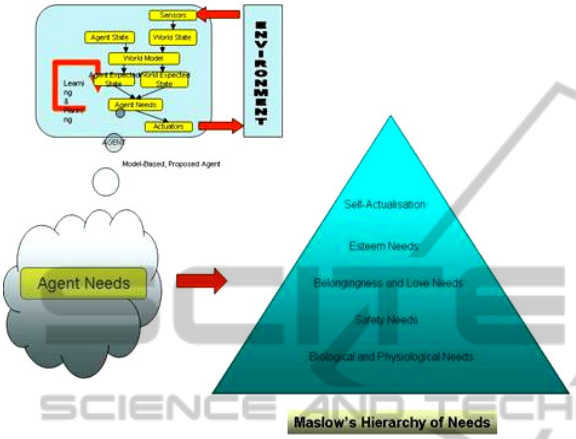

Figure 7: Maslow's Hierarchy of Needs (MHN).

Although this might not seem different from current paradigms, it actually is. Current agents observe the world and decide upon an action. What is proposed here is that an agent first observes the world and estimates what the world should look like on the next observation. This effectively assures a continuous assessment of the 'world model'.

The third difference between the needs-based model and the goal and utility-based models is the use of 'needs'. Needs are tuplets of goal and utility. For example, in a biological context, an agent may have a need of sustenance for energy. The need is a tuplet of the goal of obtaining food measured by the utility that of satisfying hunger. In a game context, the need for safety might be a tuplet of the goal of moving to a point and the utility of shelter from attack of that point. On the surface this seems simply a rewording of existing approaches. The nuance incorporated in the proposed approach is the use of a hierarchy of needs that gives representation to the relative nature of needs. It is through this use of Maslow's Hierarchy of Needs (Maslow, 1943) (MHN) that behaviour becomes modulated.

The key elements of the "needs-based" architecture are as follows. In the first place, an agent's needs are hierarchically arranged to parallel MHN. The most basic survival needs lie at the base of the pyramid. If these needs are not met, the agent can no longer function. One level above the base needs, are the needs that if not met, may threaten the base needs. Similarly, two levels above the base needs are those that may threaten the level below that, and so on up the pyramid. The implementation challenge is to draw the parallels between the human representation of MHN and the agent. This is done by expanding on the principles of 'goals' and 'utility' as shown in Figure 8 and mentioned earlier with respect to goal-based and utility-based agents.

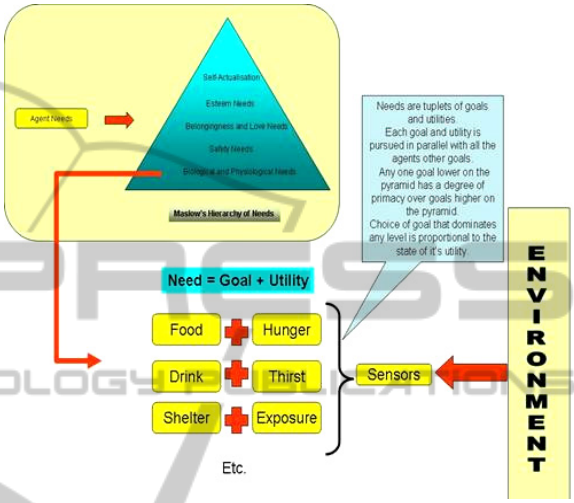

Figure 8: Agent's needs.

The MHN hierarchy cannot be considered rigid, and if used as such, it will be a dull instrument. A rigid application of MHN means rigid analysis, yet people are more complex. Indeed, the hierarchy is not fixed nor are the goals unique to a specific level. In the first case, an agent may have a 'higher purpose' that will assume primacy over even base functions. The reason for this is that there is no difference between a need that is satisfied and a need that is unattainable. While the need for safety may in principle override the need to help the group but if safety is clearly unattainable then an agent may decide to keep helping the group (i.e. "take one for the team"). Goals are also not unique to any given level. At the human level, the goal to obtain food is valid for addressing the need of hunger; however, food can also be obtained for the purpose of sharing (group belongingness) or status (obtaining rare food).

It might be asked if this is not simply a subsumption architecture (Brooks, 1991). While similar, the proposed architecture is distinct. First, the subsumption architecture speaks to the current state of the environment and the "desired external manifestations of the robot control system" (Brooks, 1986). Thus the focus is very much on what is

external to the agent. However to incorporate some sense of self (which seems necessary for any agent to successfully parallel human behaviour), it is logical that there must be some focus on internal states. For humans, the role of internal states is neatly captured by MHN. Second, subsumption architectures are implemented as finite state machines. The intent in the proposed architecture is that all levels operate in parallel and within each level multiple sub-needs are addressed. At no level is there a set number of fixed permissible states, rather there is an aggregated estimate of the degree to which needs at a given level are satisfied.

To summarize, an agent architecture has been proposed to exhibit human-like behaviour in a game environment. The proposed agent architecture is simpler than existing cognitive architectures but nonetheless comprises elements and approaches that will modulate simple reflexive behaviours. To achieve the latter, we make use of the human model of MHN and apply it to Model-based agents. It is expected that by creating an agent equivalent of MHN and implementing it within a model-based agent architecture, it will be possible to emulate human behaviour in CGAs.

3 IMPLEMENTATION OF A "NEEDS-BASED" AGENT

The paper would be incomplete without some consideration of implementation issues. Given the conceptual design of a needs-based agent that operates under the MHN framework, it is possible to make some technical judgements about the internal structure of the agent and what technologies might effect or facilitate its implementation. This section captures these early design decisions and presents our technical position for implementing a prototype "needs-based" agent.

3.1 Agent Framework

The nature of the MHN structure implies that the implementation of a needs-based agent will take the form of a multi-agent system and therefore require a multi-agent framework. Given the number of layers in the MHN and the complexity of the agent interactions within that pyramid, only the simplest of agents could be implemented as a single agent process. Also, it would be impossible to scale this implementation to more complex implementations with multiple sensors and actuators. Thus it makes sense to take the early decision that the implementation of the "needs-based" agent will be a community of agents within an agent framework. This agent framework will support the operation and management of individual agents as well as provide the essential inter-agent communications. The scope of these communications cannot be considered simple message passing, but may involve more advanced communication protocols (i.e. inter-agent negotiation or competition among agents) with specialized vocabulary requirements (ontologies). It is also vital that any agent framework used on this implementation have the flexibility to adapt to changes in the research program or allow the implementation of any requirements that haven't yet emerged. Since the purpose is not to develop a general multi-agent framework but to implement the needs-based agent architecture, one of the early decisions was to use the Java Agent Development Framework (JADE).

3.2 Java Agent Development Framework (JADE)

JADE is an open-source middleware that includes a runtime environment for JADE agents, a library of classes that programmers can use to develop agents (either directly or by tailoring the classes), and tools for administration and monitoring the activity of running agents. The important parts of the framework are the agent container, the agent management system (AMS) agent and the directory facilitator (DF) agent (Figure 9). Containers house agents and there are two types of containers: main containers and normal containers. Every main container holds an AMS agent and a DF agent in addition to any other agents and manages all the agents within a "platform". The AMS provides a naming service to ensure all agents have unique names and is the means by which agents are managed within the container. The DF provides a "yellow pages" listing of agents and the services they offer. Agents can query the DF and find other agents that offer the necessary services to achieve goals. The JADE framework follows the architectural and communications structure specified in the FIPA (Foundation for Intelligent Physical Agents) standards. FIPA is part of the IEEE Computer Society. JADE also provides features that allow the development of user-defined ontologies and complex interaction protocols.

JADE also provides essential links to future proof the implementation from the standpoint of its open-source nature, existing extensions to the JADE

platform, and existing abilities to work with other technologies. As an open source tool, the classes and framework can be tailored to address any unforeseen risk in the technical solution. The JADE community is very active and the scope of current projects and interest will ensure it continues to be so in the near future. JADE has the added advantage of existing extensions that support the development of BDI-based agents (JADEX). If the development of needs-based agents requires the implementation of Belief-Desires-Intentions algorithms, JADE has existing support. Perhaps one of the more important architectural aspects of the JADE framework is the existing work integrating JADE with OSGi technology. OSGi technology provides a mature component system designed to work across a variety of domains. The capability to work with components reduces complexity. Being able to work with established interfaces promotes reuse and creates a dynamic execution framework where agents and their services can constantly change. In other words, new agents can be added to the system while existing agents are running and OSGi will allow hot-swappable agents to be implemented. Similarly, the OSGi manages the library dependencies and prevents "Library Hell" from occurring (Library Hell is a condition where conflicts between the support libraries of the agent applications prevent the simultaneous execution of two agents. It is also called DLL Hell). Thus the JADE agent framework addresses our concerns and provides both the functionality to implement needs-based agents but also mitigates a measure of the risk.

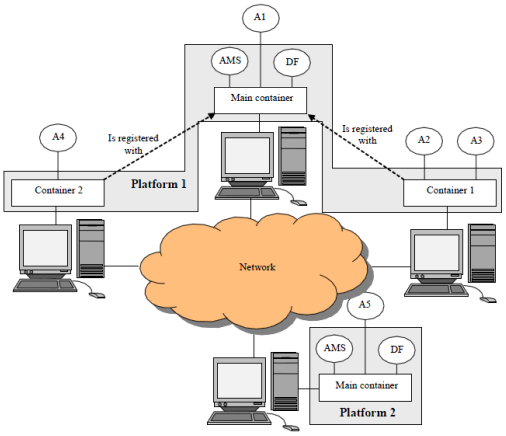

Figure 9: JADE Framework.

3.3 Agents' Types

As already discussed, the implementation of a "needs-based" agent architecture implies a multi-agent system with one or more agents operating at each layer of the MHN pyramid. As we consider the implementation, three types of agents emerge: pyramid agents, arbitration agents, and learning agents. Pyramid agents operate at one (and only one) layer of the MHN pyramid. It is possible that multiple agents will be required to effectively address a single layer. For example, a pyramid agent may model the food and hunger sensors and propose goals to acquire food. Once this decision is made, there needs to be a path for each agent in the pyramid to influence actuators. Having multiple agents attempting to grab actuators is not going to work and therefore we require some agents to act as arbitrators between conflicting goals and between competing levels of the MHN pyramid. These arbitration agents become responsible for implementing the MHN hierarchy, monitoring agent information at each level and in the end, changing actuators that will have an expected impact on the environment. Where the expected state of the agent and the expected state of the world differ from the actual state, a third type of agent needs to be involved, learning agents. Learning agents dynamically adapt the pyramid agents and arbitration agents to again bring the system to a state where predicted outcomes match environmental observations. Thus we currently foresee the requirement for three classes of agents: agents to fill a layer (or partial layer) in the MHN pyramid, agents to arbitrate between competing goals and objectives and produce actuator changes, and when prediction models fail, learning agents adapt the needs-agents to effectively address the new world order.

3.4 Design Decisions

This section captures some early design decisions and technology assessments for the implementation of a needs-based agent and makes some analysis of what the internal representation of that agent might be. At present it is possible to foresee the creation of three roles within the needs-based agent architecture. Those are pyramid roles that address elements of each layer of MHN, arbitration roles to translate conflicting goals to actuator actions, and learning nodes to dynamically modify agents when prediction fails to effectively address changes in the world state. From the infrastructure perspective, tools and technologies exist that from a first-look would allow the construction of a "needs-based" agent that operates using the MHN pyramid. The JADE agent framework has been selected as a suitable platform in that it provides a flexible multi-agent

infrastructure with extensions to other resources and technologies that may prove advantageous in future implementations. Thus, the possibility of implementing a "needs-based" agent has a measure of technical feasibility.

4 CONCLUSIONS

This paper captures the initial stages of a research project to improve the decision making performance of simulated entities in CGF software applications. By drawing the analogy between CGF synthetic entities and CGAs in games, it was determined that the current state of AI in video games is wanting, mostly due to conflicting goals (realism versus player experience). Cognitive architectures are too complex to be effectively used in multiple CGA scenarios leading to our decision to focus on agent-based solutions that emulate human behaviour without the overhead of simulating human cognition. The traditional reflex and model-based agents were found to be insufficient to emulate human behaviour effectively, so a new agent model was proposed. The "Needs-based" agent architecture is simpler than existing cognitive architectures but nonetheless comprises elements and approaches that will modulate simple reflexive behaviours.

This paper also captured some early design decisions and technologies that would allow the implementation of a "needs-based" agent. It is obvious that any implementation of a needs-based agent will be a community of smaller agents in a flexible hierarchy. This drove our decision to use JADE and the OSGi. With regard to the internal structure, we foresee the creation of pyramid agents that address the goal and utility tuplets within each level of the MHN framework; the creation of arbitration agents to address conflicting proposed actions within the model and learning agents to dynamically modify agent parameters in response to autoassociation failures. From these preliminary looks at implementation, it appears that the creation of a "needs-based" agent has a measure of technical feasibility. The next steps for the project are to proceed with the construction of a prototype to evaluate its behaviour and the feasibility of integrating such a framework across CGF applications.

REFERENCES

Anderson, J. R., 1993. *Rules of the Mind*. Hillsdale, NJ: Erlbaum.

Bourassa, M. A. J., Massey, L., 2009. Artificial Intelligence in Computer-Generated Forces - A Survey of the State-of-the-Art, Technical Memo, *DRDC Ottawa TR 2009*- In Print.

Brentano, F., et *al*., 1874. *Psychology from an Empirical Standpoint*.

Brooks, R., 1991. Intelligence Without Representation. *In Artificial Intelligence* 47. pp. 139-159.

Brooks, R., 1986. A Robust Layered Control System for a Mobile Robot. In *IEEE Journal of Robotics and Automation*, RA-2, 1. pp. 14-22.

Duch, W. et *al*., 2008. Cognitive Architectures: Where do we go from here? In *Proceedings of the 2008 Conference on Artificial General Intelligence*: Proceedings of the First AGI Conference, IOS Press, pp.122-136.

Grand., S., 2000. *Creation: Life and How to Make It*. Orion House, London.

Hawkins, J., Blakeslee, S., 2004. *On Intelligence –Owl Books*, Henry Holt and Company, New York.

Maslow, A. H., 1943. A Theory of Human Motivation, *Psychological Review* 50. pp. 370-396.

Ness, J., Tepe, V., Ritzer, D., 2004. Representing Cognition as an Intent-Driven Process. In *The Science of Simulation of Human Performance*, *Advances in Human Performance and Cognitive Engineering Research Series*, Elsevier.

Russell, S., Norvig, P., 2003. *Artificial Intelligence: A Modern Approach*. Prentice-Hall Inc. New Jersey.

Russell, S., Norvig, P., 2010. *Artificial Intelligence: A Modern Approach*. Prentice-Hall Inc. New Jersey, 3rd Edition.

Taylor, A., et al., 2009. Artificial Intelligence in Computer Generated Forces: Comparative Analysis. In *2009 Huntsville Simulation Conference*.