

# Extended Behavior Networks and Agent Personality: Investigating the Design of Character Stereotypes in the Game Unreal Tournament

Hugo da Silva Corrêa Pinto and Luis Otávio Alvares

Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)  
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil  
{hsspinto, alvares}@inf.ufrgs.br

**Abstract.** The Extended Behavior Network (EBN) is an architecture and action selection mechanism to design agents capable of selecting sets of concurrent actions in dynamic and continuous environments. It allows one to specify context-dependent motivations and build agents modularly, and has achieved good results in the Robocup and in the 3D action game Unreal Tournament. PHISH-Nets, another behavior network model capable of selecting just single actions, was applied to character modeling, with promising results. We investigate how EBNs fare on agent personality modeling via the design and analysis of 5 stereotypes in Unreal Tournament. We discuss three ways to build character personas and situate our work within other approaches. We conclude that EBNs provide a straightforward way to develop and experiment with different personalities, being interesting for building agents with simple personas and for character prototyping.

## 1 Introduction

The personality of an agent is largely characterized by its motivations and goals and how it behaves to achieve these goals. These motivations may be many and conflicting, and usually an agent has many ways of satisfying them. If the agent is situated in a dynamic domain, we have to develop a good enough action selection policy for the agent, regardless of its personality, in order to enable the agent to fulfill its goals.

Behavior Networks[1] were proposed as an action selection mechanism to choose good actions in complex and dynamic environments. They gracefully treat conflicts among the goals, are fast, robust, reactive and favor actions that contribute to more than one goal.

PHISH-Nets[2], an extension of the original behavior network[1], was proposed as an architecture to develop personalities for characters in an interactive domain, based on the Bad Wolf and Three Little Pigs tale. The main limitations of the mechanism were that it could select just one action at a time, used boolean conditions and supported only context-independent goals.

Extended Behavior Networks (EBN)[3] are another extension of [1]. They use real-valued propositions and are able to specify situation-dependent goals. Action selection is done concurrently, so more than one action may be selected

simultaneously. They have been applied with good results in the Robocup [4] [5] and in the game Unreal Tournament[6]. These works focused on validating the extended behavior network model and on assessing agent performance. To date, we are unaware of any application of EBNs to personality modeling, even though it is applicable to a larger domain than PHISH-Nets.

In this work we investigate the design of five stereotypical personalities for agents in the game Unreal Tournament [7]: The Veteran, The Novice, The Coward, The Berserker and The Samurai. Unreal Tournament is a 3D action game where warriors fight each other in an arena.

In the remaining of this paper we present an overview of extended behavior networks, followed by a description of the game environment and the designing and experimenting of each agent warrior stereotype. Next, we discuss ways to design agent personalities using EBNs based on the stereotypes presented and situate our contribution within a body of related work. We conclude by pointing the easiness of personality design with EBNs, the applicability and scope of our approach, and our next research steps.

## 2 Extended Behavior Networks

An extended behavior network can be viewed as a set of linked modules and goals that mutually inhibit and excite each other via activation spreading, starting at the goals and flowing to the modules. The modules with higher activation and executability that do not use the same resources are selected for execution at each step. In the next subsections we examine in detail the structure of the network and the action selection algorithm.

### 2.1 Structure

An extended behavior network is defined by a set of behavior modules ( $M$ ), a set of goals ( $G$ ), a set of sensors ( $S$ ), a set of resources ( $R$ ), and a set of control parameters ( $C$ ). Figure 2.1 shows the specification of part of a behavior network used in our experiments, and figure 2.2 the network built from this specification.

A goal  $i$  is defined by a proposition that must be met ( $Gi$ ), a strength value ( $Sti$ ) and a disjunction of propositions that provide the context for that goal, called the relevance condition ( $Li$ ). The strength provides the static, context-independent importance of the goal and the relevance condition provides the dynamic, context-dependent one.

The use of two kinds of conditions in the goals enables us to express goals that become more or less important, depending on the situation the agent is in.

A context independent goal is modeled leaving it without relevance conditions. Goal *EnemyHurt* in figure 2.1 is an example of such a goal. Note that a goal without relevance conditions amounts to a goal that is always relevant, i.e., its relevance is always maximal.

Each behavior module is specified by a conditions list, an effects list, an action and a resources list. The first list is a conjunction of real valued propositions that represent the needed conditions for the module to execute. The effects list is a conjunction of

propositions (each possibly negated) whose values are the values that we expect them to have after the module's action execution. The resources list is made of pairs (resource, amount), each indicating the expected amount of a resource an agent uses to perform the action.

<pre> Module precondition EnemyInSight action ShootEnemy effects EnemyHurt 0.6 LowAmmo 0.1 using Hands 2 Head 1 endModule Resource name Legs amount 2 endResource Resource name Head amount 1 endResource Resource name Hands amount 2 endResource                 </pre>	<pre> Goal condition EnemyHurt strength 0.8 context endGoal Goal condition Not LowAmmo strength 0.6 context LowAmmo endGoal Parameters name ActivationInfluence value 1.0 name InhibitionInfluence value 0.9 name Inertia value 0.5 name GlobalThreshold value 0.6 name ThresholdDecay value 0.1 endParameters                 </pre>
---	---

Fig. 1. Specification of a simple behavior network

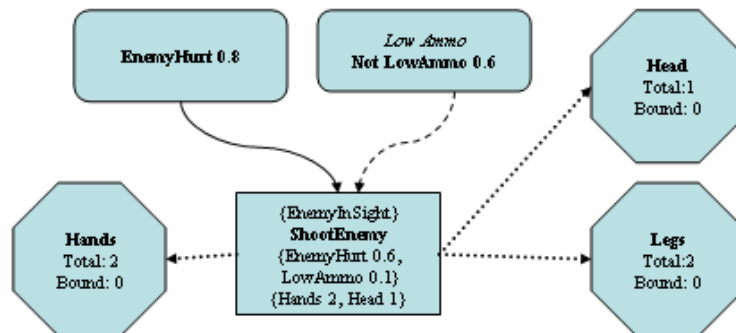


Fig. 2. Simple Behavior Network Diagram. The goals are represented by round cornered rectangles, the behaviors by sharp cornered rectangles and the resource nodes by octagons. Straight lines represent predecessor links, dashed lines conflict links and pointed lines resource links.

Goals and modules are linked with two kinds of links. Predecessor links go from a module or goal B to a module A, for each proposition in the condition list of B that is in the effects list of A, such that the proposition has the same sign (true + or false -) in both ends of the link. The link from goal *EnemyHurt* to module *ShootEnemy* in figure 2 is an example. Conflict links go from a module or goal B to a module A, for each proposition in the condition list of B that is in the effects list of A, such that the proposition has opposite signs at each end of the link. In figure 2, the link from *Not*

*LowAmmo* to *ShootEnemy* is a conflict link. Conflict links take energy away from their targets and predecessor links input energy to their targets. This way a module or goal tries to inhibit modules whose execution would undo some of its conditions and attempts to bring into execution modules whose actions would satisfy any of its conditions.

Each resource is represented by a resource node and defined by a function  $f(s)$  that specifies the expected amount of the resource available in each situation  $s$ . In addition to  $f(s)$ , each node has a variable *bound* that keeps track of the amount of bound resources and a resource activation threshold  $\theta_{\text{Re},s} \in (0..\theta]$ , where  $\theta$  is the global activation threshold. In figure 2 we see that the expected used amount of each resource is constant for all situations. This is not surprising as our agent has the same number of body parts available in any situation (the game does not account for limb loss or similar gruesome events).

The modules are linked to the resource nodes through resource links. For each resource type in the resources list of a module there is a link from the module to the corresponding resource node.

The control parameters are used to fine tune the network and have values in the range  $[0, 1]$ . The activation influence parameter  $\gamma$  controls the activation from predecessor links. Inhibition influence,  $\delta$ , the negative activation from conflict links. The inertia  $\beta$ , the global threshold  $\theta$  and the threshold decay  $\Delta\theta$  have their straightforward meanings. Their function will become clearer in the next subsection.

## 2.2 Action Selection Algorithm

The modules to be executed at each cycle are selected in the following way:

- 1) The activation  $a$  of each module is calculated.
- 2) The executability  $e$  of each module is calculated using some triangular norm operation over its condition list.
- 3) The execution-value  $h(a,e)$  is calculated by multiplying  $a$  and  $e$ . Note that this value combines the utility of executing a behavior (activation) and the probability of executing it successfully (executability). This way even modules with conditions not much satisfied may execute if they have high activation.
- 4) For each resource used by a module, starting by the last non-available resource, the module checks if it has exceeded the resource threshold and if there is enough of that resource for its execution. If so, it binds the resource.
- 5) If a module has bound all of its needed resources it executes and resets the resources thresholds to the value of the global threshold.
- 6) Each module unbinds the resources it used.

The thresholds of the resources linearly decay over time, ensuring that eventually a behavior will be able to bind its needed resources and that the most active behavior gets priority.

The formulae of Figure 3 detail the activation spreading process.

Formula (1) shows the activation that goes from a goal  $i$  to a module  $k$  through a predecessor link at instant  $t$ . Function  $f$  is a triangular norm that combines the strength

and the dynamic relevance of a goal. The term  $ex_j$  is the value of the effect proposition that is the target of a link.

$$\begin{aligned}
 a_{kg_i}^t &= \gamma \cdot f(l_{g_i}, r_{g_i}^t) \cdot ex_j & (1) \\
 a_{kg_i}^t &= -\delta \cdot f(l_{g_i}, r_{g_i}^t) \cdot ex_j, & (2) \\
 a_{kg_i}^t &= \gamma \cdot \sigma(a_{succ_{g_i}}^{t-1}) \cdot ex_j \cdot (1 - \tau(p_{succ}, s)), & (3) \\
 a_{kg_i}^t &= -\delta \cdot \sigma(a_{conf_{g_i}}^{t-1}) \cdot ex_j \cdot \tau(p_{conf}, s), & (4) \\
 a_{kg_i}^t &= \text{abs max}(a_{kg_i}^t, a_{kg_i}^t, a_{kg_i}^t, a_{kg_i}^t). & (5) \\
 a_k^t &= \beta a_k^{t-1} + \sum_i a_{kg_i}^t & (6)
 \end{aligned}$$

Fig. 3. Activation Spreading Formulae. Reproduced from [4]

Formula (2) shows the activation that goes from a goal  $i$  to a module  $k$  through a conflict link at an instant  $t$ .

Formula (3) shows the activation spreading from a module  $succ$  to a module  $k$  at an instant  $t$  through a predecessor link.  $p_{succ}$  is the proposition of the successor module

and  $a_{succ}$  the activation of the successor module.  $\tau(p_{succ}, s)$  is the value of

$p_{succ}$  in situation  $s$ . We see that the activation spreading increases as the proposition at the start of a predecessor link becomes less satisfied. Thus, we can see unsatisfied conditions as increasingly demanding sub-goals of the network. Function  $\sigma$ , shown below, is used to make the behavior modules strong attractors [8] with a high probability. This reduces unnecessary behavior switches, as small changes in the percepts will be less likely to disrupt an ongoing behavior.

$$\sigma(x) = (1 + e^{\kappa(\mu - x)})^{-1} \quad (7)$$

Formula (4) describes the activation spread from a module through a conflict link.  $a_{conf}$  and  $p_{conf}$  stand for the activation and proposition of the module that is the source of the conflict link, respectively.

Formula (6) shows that the activation of a module  $k$  at an instant  $t$  is its activation in the previous time step  $t-1$  weighted by the inertia constant  $\beta$  plus the sum of the activations retained of each goal  $i$ .

Formula (5) shows that a module retains just the activation of greatest absolute value from each goal. It amounts to keeping only the strongest path from a module to each goal.

### 3 Experiments

We designed agents with different personalities for Unreal Tournament, a 3D action game. In the game mode we used, DeathMatch, agents are warriors who must exterminate their rivals in a battle arena. Agents have many weapons available, each with certain properties (beat, pierce or explode) and several items to use. The action repertory is large (run, walk, turn, crawl, shoot, change weapons, jump, strafe, pickup item and use item among others) and an agent may carry out more than one action simultaneously, such as dodging while shooting. The scenarios are three-dimensional continuous spaces and the action happens in real-time, so the agent has to decide quickly what to do.

Five stereotypes come to mind in this scenario: The Veteran, The Novice, The Coward, The Samurai and The Berserker. The following subsections detail the requirements and design of each character.

#### 3.1 The Veteran

The Veteran is calm and rational, trying to maximize all its goals in the long run. He has great self-control and persistence and wants to kill as many enemies as possible, but never at the expense of his life. These requirements are very similar to the requirements for an agent that wants to maximize its score over a series of games. This was the case of the agent presented at [6], so we use it as a basis for the Veteran. Figure 4 shows the Behavior Network and global parameters for this character.

The overall behavior of the agent could be described as follows: It started exploring the level and kept wandering until it found an enemy (*Explore*). Upon finding an enemy it started shooting (*ShootEnemy*) and approached the enemy (*GoToEnemy*). When it reached the enemy it switched weapons and used the more powerful weapon Hammer (*FinalizeWithHammer*). After the enemy died, it stopped shooting (*StopShoot*) and started wandering again. When shot repeatedly it kept shooting and after a while stopped going to the enemy and started dodging subsequent shots. If the enemy stopped shooting it would go towards it again. When the agent was hurt in combat, if it knew the location of a medkit (*GoToKnownMedkit* had a high truth value), it would go to it and restore its health after a while. If when approaching the enemy the agent became with very low health, if there was a reachable medical kit (*MedKitReachable*) the agent would stop going to the enemy and go to the medkit, while keeping shooting unless it was close to the enemy, in which case it attempted a killing with the hammer (*FinalizeWithHammer*).

We see that this behavior matches the personality of an archetypical combat veteran: The agent is persistent when killing, heals itself when it is safe to do so and has the endurance to keep fighting even when being shot back, without panicking.

#### 3.2 The Novice

The Novice aspires to be like the veteran, has similar values, but still lacks the endurance and discipline to act properly. He is impulsive and frequently does not take the best action for a circumstance. To achieve this lower discipline and greater impulsiveness we investigated lowering two global parameters, the inertia  $\beta$  and the global threshold  $\theta$ .

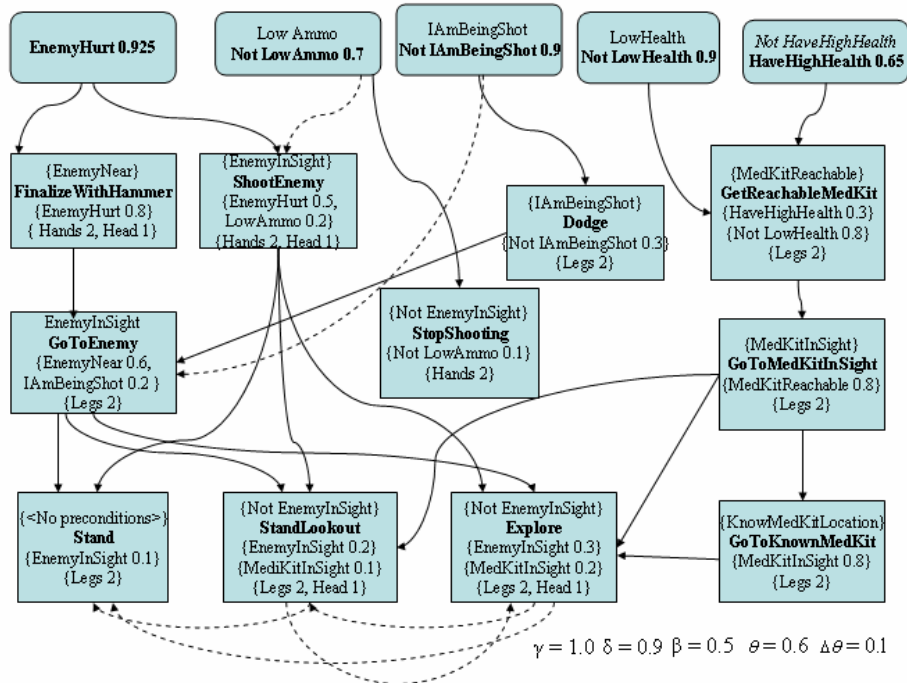


Fig. 4. Veteran's Behavior Network

We lowered the inertia to 0.1. This way the agent would be too reactive. The agent, when shot, immediately attempted to dodge. If there was a medkit nearby and it needed it, it would stop chasing the enemy and get it. Immediately dodging is not very good because dodging is not guaranteed to succeed, so an agent should dodge to avoid bursts of shots, but not single ones.

To augment the number of mistakes of the novice, we lowered the global threshold. This way we decreased the quality of action selection, as many modules surpassed the threshold simultaneously, and among modules that are over the threshold no one has priority over others. Now, often the agent shot the enemy even when it was very near and could be hammered. For the extremely low value of 0.1, the agent also often just stood still (*Stand*) instead of exploring the level (*Explore*).

The best parameters we found to bring forth the character of the Novice were  $\gamma = 1.0$ ,  $\delta = 0.9$ ,  $\beta = 0.1$ ,  $\theta = 0.25$  and  $\Delta\theta = 0.1$ .

### 3.3 The Coward

The Coward's main goal is getting out of the combat alive and unhurt, so he will avoid confrontation and will prioritize maintaining and restoring its health.

To bring forth the Coward working on the global parameters would be of little use, as he is as persistent as the Veteran and we have no reason to believe him to make

decisions less thoughtfully. Instead, we worked on the goal strengths. We lowered *EnemyHurt* and raised *HaveHighHealth*. We left the global constants untouched.

The behavior of the coward could be thus described: It started exploring the level until he found an enemy. With an enemy in sight he started shooting. When shot back, if the enemy missed him, he would start dodging after a little while, for all subsequent shots. If actually hit he would go get the medkit immediately if there was one reachable. If there was none he would keep dodging and fighting until it had a low health. When it happened he would flee combat and go restore its health, even if he had to go all the way to a far known medkit.

We see that even though the agent is far more concerned with its health and could not be described as brave anymore a key point of its specification is missing: its active avoidance of engagement. We implemented a new module for the network: *GoAwayFromEnemy*. With this module added, when the Coward spotted an enemy, he would go away from him while shooting. Figure 5 shows the full network of the Coward character. Note that adding a new behavior was a simple modular operation, dispensing adjustments.

### 3.4 The Samurai

The Samurai is cold, persistent and aggressive. To die in battle is his highest honor, and he likes fair matches. Killing his opponent is his stronger goal and he will try to achieve it even at the expense of his life. When in a fight with an enemy it won't stop to attack another agent, nor will be stopped by pain or danger.

To transform the Veteran into a Samurai we worked on the goals strengths. We set *EnemyHurt* to 1.0, *NotIamBeignShot* to 0.6, *NotLowHealth* to 0.5 and *HaveHighHealth* to 0.4. With these strengths the agent will always approach the enemy instead of dodging bullets and will not stop to get medkits if in a fight. We verified that whenever he found an enemy it went towards it shooting and then attacked with the hammer (*FinalizeWithHammer*) if the enemy had not died yet. If there was no enemy in sight the Samurai would go after medkits to restore its health. For a gamer the Samurai displayed the exact behavior we desired: He was never disturbed by pain (low health) or danger (shoots) in his pursuit of an enemy and employed good tactics (shooting from afar and hammering when near).

### 3.5 The Berserker

The Berserker is aggressive, undisciplined and non-persistent. Once in the arena he will attack fiercely its opponents, in a mad frenzy. He is insensitive to pain, and most times will not stop attacking to heal itself or even to dodge bullets.

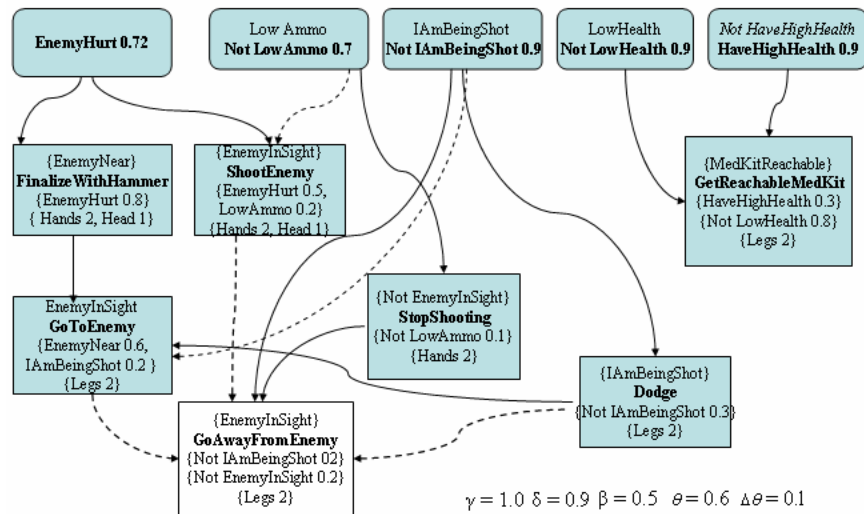
To bring into being the mad berserker we started from the Samurai. We lowered even more its sensibility to pain by decreasing the importance of goal *LowHealth* and to bring forth his frenzy we diminished both its inertia and its global threshold. Lowering  $\beta$  made the agent very reactive and lowering  $\theta$  made the agent take insane actions, such as shooting instead of hammering at close quarters.

The overall behavior of the berserker was as intended – he would not stop to dodge or heal while in combat and he fought madly, hammering and shooting everything that went into his path.

## 4 Discussion

We illustrated three approaches to develop agent personality: changing the global parameters, changing the goal strengths and changing the network topology itself.

Changing the global parameters allowed us to control two key personality characteristics: Thoughtfulness (through the activation threshold  $\theta$ ) and persistence (through the inertia  $\beta$ ).



**Fig. 4.** Part of the Coward Behavior Network. We have displayed only modules directly connected to a goal for clarity.

A high activation threshold  $\theta$  lead to better action selection as only actions that had a high activation could execute, that is, it required on average more activation spreading cycles to decide what to do next and also required higher executability for the modules. For an external observer it amounts to a thoughtful behavior, as an agent does mostly what seems effective and proper to its goals. We saw a thoughtful behavior typical of an experienced soldier in the Veteran and the thoughtless behavior of a newbie in the Novice.

A high  $\beta$  leads to a persistent behavior: An agent only changes its behavior if there is a large or long change in its sensory information. This was the case of the Veteran taking some time to start dodging bullets. A single shot was not enough to make he interrupt his course of action. Symmetrically, the Novice changed actions due to slight changes in its sensors.

The predecessor link activation constant  $\gamma$ , the conflict link activation constant  $\delta$  and the threshold decrease  $\Delta\theta$  were not used to design agent personality.

Changing the goal strengths was our first try when changing the global constants could not lead to the desired behavior. This is somewhat harder because the strength of a goal must be set in relation to the other goals of the agent. Altering the goal

strengths is the default way to alter deep personality characteristics, the very motivations and values of a character. This was the solution needed to implement both the Samurai and the Coward.

Finally for some cases we may have to add a whole new module. Adding a module to a behavior network is a straightforward operation – the network itself takes care of its integration, with the automatic creation of predecessor and conflict links. It may be a time consuming option due to subtleties that may arise in the actual implementation of the module, if one does not exist yet. If we have a library of behavior modules, then this option is also easy.

Summing all up, we could make the reverse question: how do we build agents with different personalities from scratch? First we define the agent's goals and their relative importance. Next we assemble a set of modules capable of achieving these goals. Next we tune the global parameters to achieve the subtleties of the personality. Having one working agent, making other with radically different personalities is simple, as we have seen.

## 5 Related Work

The design of agents with personality has a long tradition. Sophisticated models, with a focus on agent personality and interaction, have been developed over the last decade and the present. Usually they address the question "What is the best way to design an agent with personality and emotional traits capable of carrying out sophisticated interactions with humans and other agents?" They have shown promising results in the domains where applied, such as embodied conversational characters [9] and interactive drama [10].

Our work answers a different question: "Given that I have to design several complex agents capable of having good performance (or scores) in a real-time continuous and complex game environment in a short time span, how may I make them with different personalities?" This precludes solutions that require long processing or very complex design and favors solutions that produce a fast acceptable result. Sophisticated interaction with humans are not a concern as the interactions are quite simple and do not involve mood detection, gesture recognition or the exchange of roles.

The only previous application of behavior networks to character design that we are aware of is [2]. In this work, a behavior network model called PHISH-Nets was used to design the Big Bad Wolf and the Three Little Pigs of the famous kids tale in a simple discreet 3D environment. There was no pressure for the actions to be carried in real time. Most experiments investigated how the agent handled action failures and its capacity to improvise. Despite its interesting results for character modeling we could not use this model to answer our question, unless it was drastically modified, as for complex real-time games we need to select several actions concurrently and deal with continuous quantities.

Blumberg's [11] [12] work on synthetic characters, particularly the architecture described in [11], seems potentially fit to address the problem we pointed. It integrates learning capabilities and allows deeper emotional modeling, being more sophisticated and complicate.

Other architectures have been proposed to the modern game environment domain. Of immediate interest is an implementation of the Cog-Aff architecture [13] that used an anytime planner [14], A-UMCP. It was deployed in the Unreal Tournament domain, though for the game mode Capture the Flag. Although the Cog-Aff architecture explicitly takes into account agent personality into its design, it was not mentioned in this work [15].

The Excalibur architecture [16] is also proposed as a generic architecture for autonomous agents in complex game environments. Its distinctive feature is its ability to incorporate resources in its planning process in a sophisticated manner. Though one can easily think of ways of incorporating personality modeling into this framework we are unaware of works with this approach.

Finally we may cite the applications of the Soar architecture to computer games [17]. Soar stands in a different stratum: It is a sophisticated cognitive architecture aimed at human-level intelligence, with a considerable learning curve. Usually the foci of these works revolved around cognitive plausibility and depth of the agent models. Soar seems a good choice when one's focus is fidelity and depth but overkill for creating agents with simple personas.

## 6 Conclusion

It is relatively easy to build agents with different personalities using extended behavior networks when we consider other approaches. To build a seminal agent one starts by setting goals that reflect the character values and motivations. Next, one proceeds by assembling a set of modules capable of achieving those goals (modules that have the goal conditions as effects). Finally, by adjusting the goal strengths and global parameters one builds very different agent personalities for agents with similar capabilities.

Extended Behavior Networks are an interesting technique for building agents with similar capabilities and different personalities when we retain the focus on agent performance.

The main limitation is that the personalities built are static and simple, enabling only stereotypical personalities to come about. To be more realistic, an agent should be able to display different personas based on its mood or its emotions.

This work presented the first results of using extended behavior networks to design agent personality. Our next step is making quantitative measures of the believability of each of these personas, using ratings of human players.

## References

1. Maes, P. How to do The Right Thing. *Connection Science Journal*, Vol. 1, No. 3., 1989.
2. Rhodes, Bradley. PHISH-Nets: Planning Heuristically in Situated Hybrid Networks . MSc Thesis. MIT. 1996.
3. Dörer, K. Concurrent Behavior Selection in Extended Behavior Networks. Team Description for RoboCup2000, Melbourne. 2000.
4. Dörer, K. Extended Behavior Networks for the Magma Freiburg Team. In *RoboCup-99 Team Descriptions for the Simulation League*. Linköping University Press, 1999. p. 79-83.

5. Müller, K. Roboterfußball: Multiagentensystem CS Freiburg, Diplomarbeit. Univ. Freiburg, Germany. Feb 2001.
6. Pinto, Hugo; Alvares, L. O. C Applying Extended Behavior Networks to 3D Action Games. In: Brazilian Symposium on Games and Digital Entertainment - W Jogos, 2004, Curitiba. SBGames 2004 - Proceedings of W Jogos, 2004.
7. Unreal Tournament <http://www.unrealtournament.com> 28/03/2005.
8. Goetz, P. Attractors in Recurrent Behavior Networks. Phd Thesis. University of New York. Buffalo. 1997.
9. Cassell, J. et al. Human Conversation as a System Framework: Designing Embodied Conversational Agents, in Cassell, J. et al. (eds.), *Embodied Conversational Agents*, pp. 29-63. Cambridge, MA: MIT Press. 2000.
10. Rayes-Roth, B. Improvisational puppets, actors, and avatars, *Proceedings of Computer Games Conference*, 1996
11. Isla, D., Burke, R., Blumberg, B. et al. "A Layered Brain Architecture for Synthetic Characters," in the *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp.1051-1058, Seattle, WA, August 2001.
12. Kline, C. and Blumberg, B. The Art and Science of Synthetic Character Design. In *Proceedings of the AISB1999 Symposium on AI and Creativity in Entertainment and Visual Art*, Edinburgh, Scotland. 1999
13. Scheutz, M. and Sloman, A. A Framework for Comparing Agent Architectures. In *Proceedings UKCI'02, UK Workshop on Computational Intelligence*, Birmingham, UK. 2002.
14. Zilberstein, S. Using Anytime Algorithms in Intelligent Systems. *AI Magazine*, 17(3):73-83, 1996.
15. Hawes, Nick. An Anytime Planning Agent For Computer Game Worlds. In *Proceedings, Workshop on Agents in Computer Games at The 3rd International Conference on Computers and Games (CG'02)*, July 27th 2002. Pages 1--14.
16. Nareyek, A. Beyond the Plan-Length Criterion. In Nareyek, A. (ed.), *Local Search for Planning and Scheduling*, Springer LNAI 2148, 55-78. 2001.
17. Magerko, B. Laird, J. et al, AI Characters and Directors for Interactive Computer Games. In *Proceedings of the 2004 Innovative Applications of Artificial Intelligence Conference*, San Jose, CA, July 2004. AAAI Press.