Knowledge-Based Systems 34 (2012) 97–113

Contents lists available at SciVerse ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Improving game bot behaviours through timed emotional intelligence

Giovanni Acampora [a,*], Vincenzo Loia [b], Autilia Vitiello [b]

[a] School of Industrial Engineering, Information Systems, Eindhoven University of Technology, P.O. Box 513, 5600 MB, Eindhoven, The Netherlands
[b] Department of Computer Science, University of Salerno, 84084 Fisciano, Italy

ARTICLE INFO

Article history:
Available online 21 April 2012

Keywords:
Games
Computational intelligence
Emotions
Fuzzy control
Fuzzy markup language
Timed automata

ABSTRACT

The video game industry is a very active economic sector focusing on the design and development of entertainment applications. In this sector, different enterprises compete to design innovative video games that exploit physical and emotional capabilities of video gamers in order to achieve high levels of realism. From this point of view, video games research should be enhanced by introducing: (1) novel methodologies for modeling emotions that depict the behaviour of different characters populating a virtual environment, and (2) some abstraction technologies useful to "move" this emotional intelligence on different game platforms without additional efforts. In this paper, innovative computational intelligence techniques, like the Timed Automata based Fuzzy Controllers, have been hybridized with emotional representation methodologies in order to provide game bots with human-like capabilities and, as a consequence, improve the realism of game under design. In order to allow different competitors to exploit our "emotional engine" on different hardware platforms, the Fuzzy Markup Language (FML) has been chosen to be the main technology for designing and implementing the bots behaviour. As shown in a case study based on Unreal Tournament 2004, the game bots exploiting our approach provide a more human-likeness behaviour if compared with simple finite state automaton based bots.

1. Introduction

The entertainment industry started its golden age in 1978 with the release of *Space Invaders*, a killer-app that inspired dozens of manufacturers to enter the video games market. In particular, this game stimulated several software houses to design and develop hundreds of video games compatible with different popular consoles and computers made by pioneers of home entertainment such as Atari, Apple, Commodore, Sinclair, ColecoVision, and Mattel. These games were mainly based on elementary interactions among video gamers and a collection of simple virtual entities, being characterized by the absence of emotions and personality. The first attempt to provide these entities with unsophisticated personalities was made by *Pac-Man* programmers that designed each ghost enemy with its own distinct behaviour in order to keep the game from becoming impossibly difficult or boring to play; the real personality of all enemies has been described in more detail only at the 2011 Game Developers Conference where Pac-Man programmers stated that the red ghost would chase Pac-Man, the pink and blue ghosts would try to position themselves in front of Pac-Man's mouth,[1] whereas the orange enemy's behaviour was randomly selected.

Since golden age era, the entertainment enterprises have been trying to enhance cognitive aspects of video games and, in particular, competing for developing innovative and realistic video games that exploit both emotional and physical characteristics of video gamers. This challenge has resulted in the development of creative systems able to strongly involve video gamers inside virtual environments and, as a consequence, make video games more attractive than entertainment applications that do not take care with physical and emotional status of human players. From this point of view, video games control systems such as Nintendo Wii™, Sony EyeToy™ and Microsoft Xbox Kinect™, enhance gamers' capabilities by introducing meaningful whole body gestures and, as a consequence, accommodating a plethora of physical inputs that are strongly related to the emotional status of human player.

Nevertheless, in order to further improve the emotional features of next generation video games, some formal and efficient methodologies should be introduced so as to provide video game characters (and not only human players) with additional properties such as *personality*, *likes* and *dislikes*, *emotions*, *moods* aimed at improving their *human likeness* and *believability* [1]. However, as stated by different psychology studies, a full understanding of emotions and emotion characteristics could only be reached when their dynamic nature is taken into account. Indeed, emotions unfold over time and, depending on external as well as internal events, their intensity level may continuously change so as to generate high variability in a human behaviour [2]. As a consequence, human emotions and moods can be artificially imitated only if

* Corresponding author. Tel./fax: +39 3482786711.
E-mail addresses: g.acampora@tue.nl (G. Acampora), loia@unisa.it (V. Loia), avitiello@unisa.it (A. Vitiello).

[1] http://www.cnbc.com/id/41888021.

0950-7051/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.knosys.2012.04.012

their dynamical characteristics are opportunely analyzed. This choice will put players more deeply into the game story, make them feel alive and, consequently, create entertainment scenarios characterized by high level of realism.

In order to achieve this goal, this paper proposes a novel cross-platform architecture (whose preliminary version is presented in [3]) meant to model Non-Player Characters (NPCs) behaviour by taking into account different features such as emotions, personality and dynamic selection of actions, and strongly improving NPCs' human-likeness. This result has been mainly achieved by merging some theories from psychology area, OCC [4] and OCEAN [5], together with an innovative computational intelligence technique named Timed Automata based Fuzzy Controllers (TAFCs) [6,7] that, unlike conventional fuzzy systems, models inference engine whose performance is strongly depending upon temporal concepts. OCC and OCEAN are simultaneously used to model the personality and the collection of emotions that characterize the behaviour of a given NPC, whereas, TAFCs are used as a decision making system able to analyze the video game virtual environment and the NPCs' emotional state in order to select the most suitable actions improving NPCs human likeness at a given time. Our proposal represents the first attempt to put together relevant concepts, such as time, emotions and personalities, which strongly influence the human being's behaviour, in order to define a so-called *timed emotional intelligence* useful to enhance video game bots from a human likeness point of view. This hybrid approach has been developed by using the Fuzzy Markup Language (FML) [8], a computer language useful for implementing fuzzy systems in a hardware-independent way. This choice will allow different competitors to implement the proposed approach for modeling bot behaviours on different game platforms (such as Sony PS3, Microsoft Xbox 360, and Nintendo Wii) in a simple and direct way.

Our approach for cross-platform timed emotional intelligence has been tested by interfacing it with a collection of tools, such as UnrealEngine, GameBots 2004 and Pogamut, that are usually used for the design of video games bots. In particular, a framework capable of generating a network of NPCs and monitoring and evaluating their matches has been developed. In order to estimate the human likeness obtained by the proposed approach, a Turing test has been performed, since, even if it was not intended to, Turing's test is considered as a reference for believability evaluation [9].

2. Related works

In the last years, several efforts have been led in order to model video game bots with a human-like behaviours and to achieve a greater enjoyment in playing video games. In particular, in [10], in order to design interesting opponents, the authors propose to teach a computer opponent to play like a human using machine learning techniques. Similarly, in [11], the authors introduced *Dynamic Scripting*, an online learning technique based on the use of reinforcement learning to adjust a mechanism for selection to choose between various scripted behaviours. The stochastic selection mechanism and the online learning provided the play with a degree of unpredictability so to make it more human-likeness. In addition, in [12], the authors studied a Bayesian-based approach to the derivation and imitation of human strategic behaviour and motion patterns in commercial computer games. They demonstrated the effectiveness of their approach in producing convincingly human-like game agents in conjunction with the believability-testing system. Finally, in [13], the authors propose a method for generating natural-looking behaviours for virtual characters using a data-driven method called behaviour capture. In details, they describe the techniques for capturing trainer-generated traces, generalizing these traces and using the traces to generate behaviours during game-play. The work presents several trace generalization mechanisms, including a technique that learns a Hidden-Markov Model (HMM).

However, all these works do not deal with emotions, which are an essential component of the believability of embodied characters interacting with humans [14–16]. Therefore, in order to provide bots with a more believability and induce a player to participate in the game story more deeply so to transform a game into an "intense emotional journey" [17], the last researches have been studying computational models to include emotions and personality in the bot behaviours. In particular, in [18], the authors describe the design and implementation of a module of emotions and personality for synthetic actors. In details, the designed model is implemented using fuzzy logic, Finite State Machines (FSMs), and probability theory. The functionalities of the module were shown using a demo version implemented by the videogame engine Unreal®2 Runtime. Another example is provided in [19], where the authors propose a kind of robotics inspired behavioural AI techniques to simulate characters' personalities in a commercial video game.

Although the above said works improve the state of the art related to the modeling of NPCs, they still suffer from a lack of temporal concepts management which strongly influences the emotional state of an intelligent entity. Indeed, let us consider a human being under stress and let T be a timer that counts the number of minutes during which this human being works under stress; it is clear that the human beings actions will change when T and stress intensity grow. This example shows how intelligent entity's actions depend on a combination of emotions and time.

Therefore, in order to overcome this temporal weakness, our idea is to exploit dynamic fuzzy engines, named TAFCs, capable of modelling NPCs behaviour in a fuzzy and time-dependent way by analyzing emotions and personality of the intelligent entity under design.

3. Background knowledge: OCC/OCEAN approaches, TAFCs and FML

The goal of this paper is to propose a novel cross-platform architecture enhancing the modeling of Non-Player Characters (NPCs) behaviour by taking into account different features such as emotions, personality, and dynamic selection of actions. This aim is achieved by merging some theories from psychology, like OCC and OCEAN, together with an innovative computational intelligence technique named Timed Automata based Fuzzy Controllers (TAFCs). Moreover, our proposal has been developed in a hardware-independent way through the exploitation of an XML-based computer language, called Fuzzy Markup Language (FML).

Therefore, in this section, before presenting our approach in a detailed way, a description of the OCC/OCEAN approaches, TAFCs and FML is given.

3.1. OCC/OCEAN approaches

There are several emotion models available in literature [22,23]. However, the so-called OCC model developed by Ortony, Clore and Collins has established itself as a standard model for the emotion synthesis [16]. This model considers emotions as valenced reactions to three kinds of stimuli: *consequences of events*, *actions of agents*, and *aspects of objects*. These stimuli are appraised according to individual's goals, standards and attitudes. Specifically, an individual judges the following [24]:

- the desirability of an event, i.e., the congruence of its consequences with individual's goals (an event is *pleasant* if it helps the individual to reach his/her goal, and *unpleasant* if it prevents him/her from reaching his goal);

- the approbation of an action, i.e, its conformity to norms and standards;
- the attraction of an object, i.e. the correspondence of its aspects with individual's likings.

A further differentiation consists in the fact that events can have consequences for others or for oneself and that an acting agent can be another or oneself. The consequences of an event for another can be divided into desirable and undesirable; the consequences for oneself as relevant or irrelevant expectations. Relevant expectations for oneself finally can be differentiated again according to whether they actually occur or not (confirmed/disconfirmed) [25]. This differentiation leads to a structure of emotion types here shown in Fig. 1 [16].

The advantage of this theory is to be very close to a computational approach and so central to most of models of emotions. However, this theory is characterized by the negative aspect of not being capable of defining intensity of final emotions to launch [26]. For this reason, over the years there has been a growing development of emotional architectures based on the OCC emotion representation, such as FLAME [27] and ParleE [28], designed for the computation of emotional state of an intelligence entity. In particular, our proposal to calculate the emotional state of a NPC, as described in subsection 4.1, is inspired to these existing approaches.

Also the identification of traits and structure of human personality has been one of the most relevant research goals in psychology field. One of the most known model for personality representation is the Five Factor Model, which describes the human personality as composed of five broad dimensions: *openness*, *conscientiousness*, *extraversion*, *agreeableness*, and *neuroticism*. These five factors are also referred to by the common acronym "OCEAN". In details, the OCEAN factors are characterized by the following constituent traits:

- *Openness* characterizes people admiring/appreciating for art, uncommon ideas, adventure, curiosity.
- *Conscientiousness* results in a planned behaviour which characterizes people inclined to self-discipline and attainment of personal objectives.
- *Extraversion* characterizes people with positive emotions, vitality and aptitude to seek stimuli from others.
- *Agreeableness* characterizes people with an aptitude to be compassionate and collaborative rather than suspicious and antagonistic towards others.
- *Neuroticism* characterizes people with an inclination to experience unpleasant emotions easily, such as anger, anxiety and depression.

3.2. Timed Automata based Fuzzy Controllers

Timed Automata based Fuzzy Controllers (TAFCs) are evolvable fuzzy inference engines highly suitable for modeling dynamic systems thanks to their capability of modifying themselves during their life-cycle. More in detail, a TAFC lives a sequence of discrete temporal periods named *control eras*. During each control era, the system's behaviour is modelled by means of an opportune fuzzy controller which represents the so-called *control configuration* of the TAFC. In order to allow a switching between adjacent control eras, a mechanism named *control time* is needed. Specifically, the control time sets the current control configuration, which models the current behaviour of a system and the duration of control eras.

TAFCs implement new concepts of control era, control configuration and control time by means of a timed automaton. In particular, TAFCs manage the control eras (and related control configurations) by associating each of them with a state in the timed automaton. The control eras progression can be determined by exploiting the automaton *run* concept (Definition 4). Indeed, the ith discrete transition can be used to throw a temporal event which moves the TAFC from the ith control era to the $(i+1)$th one.

However, it is necessary to extend the typical timed automaton (Definition 3) by modifying the definition of timed automaton

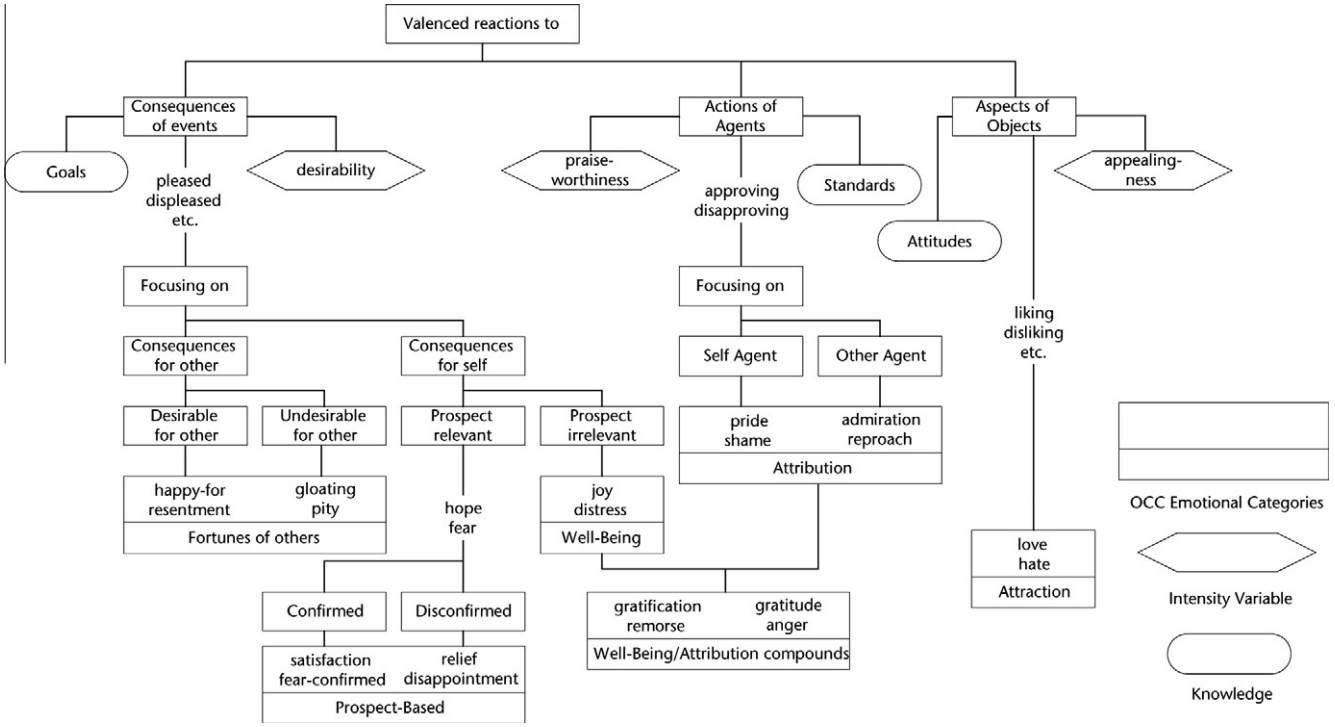

Fig. 1. The OCC model.

transition. Indeed, since each automaton's state represents a system's control era with the corresponding control configuration, then, its outgoing transitions would have to be able to transform control configurations. In order to define this task, timed automata transition definition has been extended with a sequence of *transformation operators*.

Once the automaton computation starts over a given timed word, the state transitions will opportunely modify the current control configuration in the following one. In this vision, a timed word determines how and when to execute the switching among following control eras. So, a timed word coincides with the control time concept.

Therefore, a TAFC, as formally defined in the last subsection, is a couple of two components: an extended timed automaton that describes the dynamic evolution of a system and a traditional fuzzy controller modeling the control behaviour of the system during first phase of its existence.

Hereafter, before formally describing TAFCs, the formal definition of the classical timed automata [29] is given.

3.2.1. Timed automata

A timed automaton is a standard finite-state automaton extended with a finite collection of real-valued *clocks* providing a straightforward way to represent time related events, whereas automata-based approaches cannot offer this feature. The clocks can be reset to 0 (independently from each other) with the automaton transitions, and keep track of the time elapsed since the last reset. The transitions of a timed automaton are labeled with a *guard* (a condition on clocks), an *action* or *symbol* on alphabet Σ, and a *clock reset* (a subset of clocks to be reset). The set of behaviours expressed by a system modeled by means of a timed automaton is defined by a *timed language*, i.e., a collection of *timed words*. Both timed concepts are defined as follows.

Definition 1. A *time sequence* $\tau = \tau_1\tau_2\ldots$ is an infinite sequence of time values $\tau_i \in \mathbb{R}$ with $\tau_i > 0$, satisfying the following constraints:

1. *Monotonicity*: τ increases strictly monotonically; that is, $\tau_i < \tau_{i+1}$ for all $i \geqslant i+1$.
2. *Progress*: For every $t \in \mathbb{R}$, there is some $i \geqslant 1$ such that $\tau_i > t$.

A *timed word* over an alphabet Σ is a pair (σ, τ) where $\sigma = \sigma_1\sigma_2\ldots$ is an infinite word over Σ and τ is a time sequence. A *timed language* over Σ is a set of timed words on Σ.

Definition 2. For a set X of clock variables, the set $\Phi(X)$ of *clock constraints* δ is defined inductively by

$$\delta := x \leqslant c | c \leqslant x | \neg\delta | \delta_1 \wedge \delta_2$$

where x is a clock in X and c is a constant in $\mathbb{Q}$, the set of nonnegative rationals.

A clock interpretation v for a set X of clocks assigns a real value to each clock; that is, it is a mapping from X to $\mathbb{R}$. We say that a clock interpretation v for X satisfies a clock constraint δ over X iff δ evaluates to true using the values given by v. For $t \in \mathbb{R}$, $v + t$ denotes the clock interpretation which maps every clock x to the value $v(x) + t$, and the clock interpretation $t \cdot v$ assigns to each clock x the value $t \cdot v(x)$. For $Y \subseteq X$, $[Y \mapsto t]v$ denotes the clock interpretation for X which assigns t to each $x \in Y$, and agrees with v over the rest of the clocks.

Now, a precise definition of timed transition table, which determines the timed automaton behaviour, is given:

Definition 3. A *timed transition table* $\mathcal{A}$ is a tuple $\langle \Sigma, S, S_0, C, E \rangle$, where

- Σ is a finite alphabet,
- S is a finite set of states,
- $S_0 \subseteq S$ is a set of start states,
- C is finite set of clocks, and
- $E \subseteq S \times S \times \Sigma \times 2^C \times \Phi(C)$ gives the set of transitions. An edge $\langle s, s', a, \lambda, \delta \rangle$ represents a transition from state s to state s' on input symbol a. The set $\lambda \subseteq C$ gives the clocks to be reset with this transition, and δ is a clock constraint over C.

If (σ, τ) is a timed word viewed as an input to an automaton, it presents the symbol σ_i at time τ_i. If each symbol σ_i is interpreted to denote an event occurrence then the corresponding component τ_i is interpreted as the time of occurrence of σ_i. Given a timed word (σ, τ), the timed transition table $\mathcal{A}$ starts in one of its start states at time 0 with all clocks initialized to 0. As time advances, the values of all clocks change, reflecting the elapsed time. At time τ_i, $\mathcal{A}$ state from s to s' using some transition of the form $\langle s, s', \sigma_i, \lambda, \delta \rangle$ reading the input σ_i, if the current values of clocks satisfy δ. With this transition the clocks in λ are reset to 0, and thus start continuing time with respect to the time of occurrence of this transition. Formally, this timed behaviour is captured by introducing *runs* of timed transition tables.

Definition 4. A run r, denoted by $(\bar{s}, \bar{v})$, of a timed transition table $\langle \Sigma, S, S_0, C, E \rangle$ over a timed word (σ, τ) is an infinite sequence of the form

$$r : \langle s_0, v_0 \rangle \xrightarrow[\tau_1]{\sigma_1} \langle s_1, v_1 \rangle \xrightarrow[\tau_2]{\sigma_2} \langle s_2, v_2 \rangle \xrightarrow[\tau_3]{\sigma_3} \ldots$$

with $s_i \in S$ and $v_i \in [C \to \mathbb{R}]$, for all $i \geqslant 0$, satisfying the following requirements:

- *Initiation*: $s_0 \in S_0$ and $v_0(x) = 0$ for all $x \in C$.
- *Consecution*: for all $i \geqslant 1$, there is an edge in E of the form $\langle s_{i-1}, s_i, \sigma_i, \lambda_i, \delta_i \rangle$ such that $(v_{i-1} + \tau_i - \tau_{i-1})$ satisfies δ_i and v_i equals $[\lambda_i \mapsto 0]\,(v_{i-1} + \tau_i - \tau_{i-1})$.

3.2.2. TAFC formal definition

The first step towards the formal definition of a TAFC is to introduce a collection of operators capable of changing control configurations. In order to define the so-called *transformation operators*, the labeled tree representation [36] of a standard fuzzy controller is exploited. Since a labeled tree is a connected, acyclic labeled graph, the use of labeled tree representation is advantageous because the operations (Insert, Delete and Update) changing a fuzzy controller, defined by means of a labeled tree, are simple, flexible and computationally efficient.

In detail, transformation operators will change a TAFC's control configuration modeled through a labeled tree executing the following operations: adding ($\oplus$) or deleting ($\ominus$) a variable; adding ($\boxplus^k$), removing ($\boxminus^k$) or changing ($\boxdot^k$) k rules in the rule base; changing implication method of the rule base ($\sim$); adding ($\circledast$), deleting ($\oslash$) or changing ($\circledcirc$) a term to a variable; changing defuzzify method ($\propto$), aggregation method ($\bowtie$) or default value ($*$) of an output variable; changing lower bound ($\ltimes$) or upper bound ($\rtimes$) of the universe of discourse of a variable. Besides, other four operators are defined independently. In details, the first one does not concern with changes to the fuzzy controller structure because it sets frequency sampling ($\triangle$), whereas the others deal with a complete replacement of a fuzzy controller executing these operations: returning to the initial control configuration ($\dagger$); setting control configuration to that of the destination state ($\ddagger$) or transforming a control configuration in itself ($\curlyvee$).

After describing all transformation operators, it is possible to define the collection of transformation operators C_{op}.

Definition 5. The set of transformation operators acting on a control configuration is

$$C_{op} = \{\oplus, \ominus, \boxplus^k, \boxminus^k, \boxdot^k, \sim, \ltimes, \rtimes, \bowtie, \circledast, \oslash, \circledcirc, \divideontimes, \triangle, \propto, \dagger, \ddagger, \curlyvee\}$$

Once the set of transformation operators C_{op} has been introduced, it is necessary to redefine the Timed Automaton concept in order to consider a novel kind of transition edges capable of changing the control configuration of the modeled system. In particular, the standard transitions set of timed automata E is replaced by the following:

$$E_C \subseteq S \times S \times \Sigma \times 2^C \times \Phi(C) \times C_{op}^*$$

where C_{op}^* represents the set of all possible sequences of transformation operators, i.e, $C_{op}^* = \cup_{n \geqslant 1} C_{op}^n$ where C_{op}^n is the set of all possible sequences of n operators with $n \geqslant 1$. Now, it is possible to provide an extended definition of a timed automaton:

Definition 6. A *timed control transition table* $\mathcal{A}_C$ is a tuple $\langle \Sigma \cup \{\epsilon\}, S, S_0, C, E_C \rangle$, where

- Σ is a finite alphabet;
- ϵ represents empty event, that is, when it is on a transition, the crossing of this transition only depends on temporal constraints;
- S is a finite set of states;
- $S_0 \subseteq S$ is a set of start states;
- C is finite set of clocks;
- $E_C \subseteq S \times S \times \Sigma \times 2^C \times \Phi(C) \times C_{op}^*$ gives the set of transitions. An edge $\langle s, s', a, \lambda, \delta, o^n \rangle$ represents a transition from state s to state s' on input symbol a which can be also the empty event. The set $\lambda \subseteq C$ gives the clocks to be reset with this transition, δ is a clock constraint over C and $o^n \in C_{op}^*$ is a sequence of n transformation operators, with $n \geqslant 1$, defined in order to change the current control configuration of modeled system.

In each sequence of transformation operators an operator can be repeated in order to execute the same task on different system's components (e.g., to modify the universe of discourse of different variables). Moreover, it is important to establish that the operators are executed in the same order as their definition in the sequence (Definition 8).

Definition 7. Let F be a classical fuzzy controller modelled using a labeled tree and let $o \in C_{op}$ be a transformation operator, then $G = o(F)$ is the fuzzy controller obtained to apply the operator o on fuzzy controller F.

Definition 8. Let F be a fuzzy controller modelled using a labeled tree and let $o^n \in C_{op}^*$ be a sequence $o^n = (o_1, o_2, \ldots, o_n)$ where $o_i \in C_{op}\ \forall i \in \{1, 2, \ldots, n\}$ then $G = o^n(F) = (o_n(o_{n-1}(\ldots(o_2(o_1(F))))))$ is the fuzzy controller obtained to apply the operators $o_1, o_2, \ldots, o_{n-1}, o_n$ on fuzzy controller F in the same order as listed in the sequence o^n.

At this point, it is possible to give a formal definition of a TAFC and the properties characterizing its dynamic behaviour.

Definition 9. A TAFC T is an ordered pair composed of an initial control configuration, represented by a fuzzy controller named F^0, together with a timed control transition table T_C. Formally:

$$T = (F^0, T_C).$$

The TAFC properties which define the dynamic behaviour of a system are: *control evolution* and *control run*.

In the following definitions we will use the subsequent notation: the symbols s_i and z_i ($i \in \mathbb{N}$) represent, respectively, the states of an automaton defined by means of the Definition 6 and the transformation operators sequences defined in the collection C_{op}^*; at the same time the symbols s^j and z^j ($j \in \mathbb{N}$) represent an infinite collection of variables that assume respectively the value of states s_i and z_i.

The control evolution is a mapping among states S contained in T_C and the collection of possible control configurations obtained starting from F^0. More in detail, the control evolution is a mathematical succession, generated in an inductive way, which maps each state in S with a one or more control configurations obtained by sequentially applying over F^0 the transformation operators in $S \times S \times \Sigma \cup \{\epsilon\} \times 2^C \times \Phi(C) \times C_{op}^*$. Then:

Definition 10 (*Control evolution*). Let $T = (F^0, T_C)$ be a TAFC defined over a timed control transition table $\langle \Sigma \cup \{\epsilon\}, S, S_0, C, E_C \rangle$ with $S = \{s_0, s_1, \ldots, s_{|S|-1}\}$ the finite set of automaton states; let F^* be the collection of all possible fuzzy controllers modelled by means of the labeled tree idea; let $\Omega = \{z_1, z_2, \ldots, z_{|\Omega|}\}$ be a subset of ordered sequences in C_{op}^*, that is, $z_i = (o_1, o_2, \ldots, o_{|z_i|})\ \forall i \in \{1, 2, \ldots, |\Omega|\}$, employed to define the edges in E_C. Then, the control evolution Ψ over a state $s^0 \in S_0$ is:

$$\Psi : \mathbb{N} \to S \times F^*$$

defined inductively, as follows:

The base case ($i = 0$). Let $s^0 \in S_0$ be an initial state of $\langle \Sigma \cup \{\epsilon\}, S, S_0, C, E_C \rangle$, then:

$$\Psi(0) = (s^0, F^0)$$

The inductive step ($i > 0$). Let $\Psi(i-1)$, with $i > 1$, be defined as:

$$\Psi(i-1) = (s^{i-1}, F^{i-1})$$

where $s^{i-1} \in S$ and $F^{i-1} \in F^*$, then:

$$\Psi(i) = (s^i, F^i)$$

with $s^i \in S$, $F^i = z^i(F^{i-1})$, $z^i \in \Omega$ and $\langle s^{i-1}, s^i, a, \lambda, \delta, z^i \rangle \in E_C$.

More intuitively, the expression. (1) shows the sequence of pairs composing a control evolution over $s^0 \in S_0$ together with the fuzzy transformations obtained by exploiting the z^i sequences of operators.

$$\begin{array}{lcl} \Psi(0) : s^0 \in S_0 & \to & F^0 \\ & & \downarrow z^1 \\ \Psi(1) : s^1 \in S & \to & F^1 \\ & & \downarrow z^2 \\ \Psi(2) : s^2 \in S & \to & F^2 \\ & & \downarrow z^3 \\ \vdots & & \vdots \\ & & \downarrow z^j \\ \Psi(j) : s^j \in S & \to & F^j \\ & & \downarrow z^{j+1} \\ \vdots & & \vdots \end{array} \tag{1}$$

The image of function Ψ, I_Ψ, can be finite or infinite. This depends upon the topology of graph modeling the component T_C of the TAFC.

Obviously, the control evolution only represents a mapping between states of timed automaton T_C and collection of control configurations, computable starting from F^0 by applying different sequences of operators in Ω; no dynamic aspects are considered

in the control evolution definition and, therefore, it is necessary to introduce the idea of control run, extending the initial idea of run of standard timed transition table (Definition 4).

Definition 11. Let Ψ be a control evolution, then a control run r_c, denoted by $(\bar{s}, \bar{v})$, of a timed transition table $\langle \Sigma \cup \{\epsilon\}, S, S_0, C, E_C \rangle$ over a timed word (σ, τ) and a collection of sequences of transformation operators $\Omega = \{z_1, z_2, \ldots, z_{|\Omega|}\} \subseteq C_{op}^*$, is an infinite sequence of the form

$$r_c : \langle s^0, v_0 \rangle \xrightarrow[\tau_1]{\sigma_1, z^1} \langle s^1, v_1 \rangle \xrightarrow[\tau_2]{\sigma_2, z^2} \langle s^2, v_2 \rangle \xrightarrow[\tau_3]{\sigma_3, z^3} \ldots$$

with $s^i \in S$ and $v_i \in [C \to \mathbb{R}]$, for all $i \geqslant 0$, and $z^i \in C_{op}^*$, for all $i \geqslant 1$, satisfying the following requirements:

- *Initiation*: $s^0 \in S_0$ and $v_0(x) = 0$ for all $x \in C$.
- *Consecution*: for all $i \geqslant 1$, there is an edge in E_C of the form $\langle s^{i-1}, s^i, \sigma_i, \lambda_i, \delta_i, z^i \rangle$ such that $(v_{i-1} + \tau_i - \tau_{i-1})$ satisfies δ_i and v_i equals $[\lambda_i \mapsto 0](v_{i-1} + \tau_i - \tau_{i-1})$.
- *Atomicity*: The operators of sequence $z^i \in C_{op}^*$ are atomic operations and their computation time is equal to 0, i.e, they do not modify the duration of permanence in the automaton state s^i, $(\tau_i - \tau_{i-1})$.
- *Evolution*: each state s^i of a pair $\langle s^i, v_i \rangle$ in r_c is mapped on a FLC F^i as described by the control evolution Ψ.
- *Discrete Progression*: the clocks progression is performed in a discrete way by means of the so-called *control time unit*, $u \in \mathbb{Q}$. Specifically, u is the time amount between an instant t_1 and the following one t_2 and so it is also the time amount for the clocks increase.

If $T = (F^0, T_C)$ is a TAFC which models a given system, then the set of control runs r_c defined over the timed language L, generated by T_C, completely describes the collection dynamic behaviours of the system, whereas, the control run r_c defined over a single word $w_i \in L$ defines a precise dynamic behaviour of the system, so w_i defines the Control Time.

Definition 12 (*Control time*). If $T = (F^0, T_C)$ is a TAFC and T_C is a timed automaton recognizing the timed language $L = w_1, w_2, w_3, \ldots, w_i, \ldots$ and w_i is a timed word and r_c is a control run defined over w_i then w_i is a Control Time of the system.

Finally, it is possible to give a formal description of control era and control configuration concepts.

Definition 13 (*Control era and control configuration*). If r_c is a control run defined over the Control Time $w_i = (\sigma, \tau) \in L$:

$$r_c : \langle s^0, v_0 \rangle \xrightarrow[\tau_1]{\sigma_1, z^1} \langle s^1, v_1 \rangle \xrightarrow[\tau_2]{\sigma_2, z^2} \langle s^2, v_2 \rangle \xrightarrow[\tau_3]{\sigma_3, z^3} \ldots$$

then time interval between the instant τ_i and τ_{i+1} is the ith control era of system and the FLC F^i which depicts the system during the same interval is defined as the ith control configuration.

3.3. Fuzzy markup language

Fuzzy Markup Language (FML) is an emerging XML-based markup language whose main aim is the design and implementation of fuzzy controllers (FLC). In particular, it is used to model two well-known kinds of fuzzy controllers: Mamdani and Takagi–Sugeno–Kang (TSK). FML programs are coded through a collection of correlated semantic tags, capable of modeling the different components of classical fuzzy controllers by exploiting abstraction benefits offered by XML tools. Differently from other similar approaches, used to describe fuzzy controllers such as Fuzzy Control Language[2] (FCL) or MATLAB Fuzzy Inference System (FIS) developed by the Mathworks[3], FML allows fuzzy designers to simply code their ideas on heterogeneous hardware, without understanding details related to different platforms, but by enabling to define FLCs characterized by an additional property: *transparency*. It means that, thanks to FML, it is possible to implement the same FLC on different hardware architectures with a minimal effort and without additional design and implementation steps. In short, thanks to FML it is possible to model a fuzzy controller in a human-readable and hardware independent way. In detail, FML is essentially composed of three layers:

- XML in order to create a new markup language for fuzzy logic control;
- a XML Schema in order to define the legal building blocks;
- extensible stylesheet language transformations (XSLT) in order to convert a fuzzy controller description into a specific programming language.

Fig. 2 gives a sample of FML code modelling a fuzzy controller characterized by the fuzzy concept *temperature*. As shown in the figure, FML grammar defines a tag for modelling fuzzy concepts, fuzzy rules and fuzzy inference engines. For example, the FML tag ⟨FuzzyController⟩ is used to open any FML program, whereas, the tags ⟨KnowledgeBase⟩ and ⟨Rulebase⟩ are used, respectively, to model the set of fuzzy concepts and the set of fuzzy rules. Each tag is characterized by some attributes. For example, the tag ⟨FuzzyVariable⟩ has the following attributes: name, scale, domainLeft, domainRight, type and, only for output concept, accumulation, defuzzifier and defaultValue. The *name* defines the name of fuzzy concept; the *scale* is used to define the scale used to measure the fuzzy concept; *domainLeft* and *domainRight* are used to model the universe of discourse of fuzzy concept, i.e, the set of real values related to fuzzy concept; the position of fuzzy concept into rule (consequent part or antecedent part) is defined by attribute *type* (input/output); the *accumulation* defines the method of accumulation that is a method that permits the combination of results of a variable of each rule in a final result; the *defuzzifier* defines the method used to execute the conversion from a fuzzy set, obtained after an aggregation process, into a numerical value to give it as an output to the system; the *defaultValue* defines a real value only used when no rule has fired for the variable at issue.

Once the fundamental components of our proposal have been described, in the next section, the architecture of our framework will be defined.

4. A cross-platform artificial mind for video games non-player characters

In order to improve the current video game solutions through the implementation of bots capable of showing a human-likeness behaviour, a new framework for behaviours representation, called *Timed and Emotional Artificial Mind for NPCs* (TEAM), is introduced. Differently from other approaches (see Section 2), our proposal attempts to model NPCs behaviours by taking into account, simultaneously, two different concepts strongly influencing the human beings behaviour: emotions and time. In this vision, each game bot is characterized by a collection of emotions and a subjective personality that, unlike other proposed approaches, are evaluated in a time dependent way in order to provide NPCs with a collection of behaviours similar to those which characterize human beings. Indeed, in the same way as humans, game bots modelled through our framework will make decisions by considering their emotional

[2] http://www.fuzzytech.com/binaries/ieccd1.pdf.

[3] http://www.mathworks.com/.

```
<FuzzyController ip=''127.0.0.1'' name=''FuzzySystem''>
  <KnowledgeBase ip = ''127.0.0.1''>
    <FuzzyVariable name=''temperature''
        domainleft = ''0'' domainright = ''40''
        scale = ''Celsius'' type = ''input''>
      <FuzzyTerm name=''low''>
        <TriangularShape Param1 = ''0.0'' Param2 = ''15.0'' Param3 = ''25.0''/>
      </FuzzyTerm>
      <FuzzyTerm name=''medium''>
        <TriangularShape Param1 = ''15.0'' Param2 = ''25.0'' Param3 = ''35.0''/>
      </FuzzyTerm>
      <FuzzyTerm name=''high''>
        <TriangularShape Param1 = ''25.0'' Param2 = ''33.0'' Param3 = ''40.0''/>
      </FuzzyTerm>
    </FuzzyVariable>
    .......
  </KnowledgeBase>
  <RuleBase ip=''127.0.0.1''
        name=''RuleBase1'' activationMethod=''MIN''
        andMethod=''MIN'' orMethod=''MAX'' type=''mamdani''>
    ......
  </RuleBase>
</FuzzyController>
```

Fig. 2. FML sample program.

status, their own personality and their time perception. In particular, this approach allows bot's emotions and mood to be associated with a variable fuzzy value, whose intensity is depending upon the time. By reasoning in this way, our bots will exhibit a different behaviour depending on the way they show anger for a minute or for 3 h.

In order to achieve this level of realism, the proposed framework combines a novel fuzzy controller methodology, named *Timed Automata based Fuzzy Controllers (TAFCs)*, together with theories from psychology like *Orthony, Clore and Collins(OCC)* and *OCEAN* in order to improve the state of the art related to the modeling of NPCs (see Section 2). In particular, our hybrid approach allows to:

- analyze emotions, personality and physical conditions of an intelligent entity by means of OCC and OCEAN representation;
- make autonomous decisions by considering temporal concepts, which strongly influence emotional states, by exploiting TAFCs. Indeed, thanks to their dynamic features, TAFCs are capable of overcoming the temporal weakness of Finite state machines (FSMs) typically used by the existing approaches [20,21].

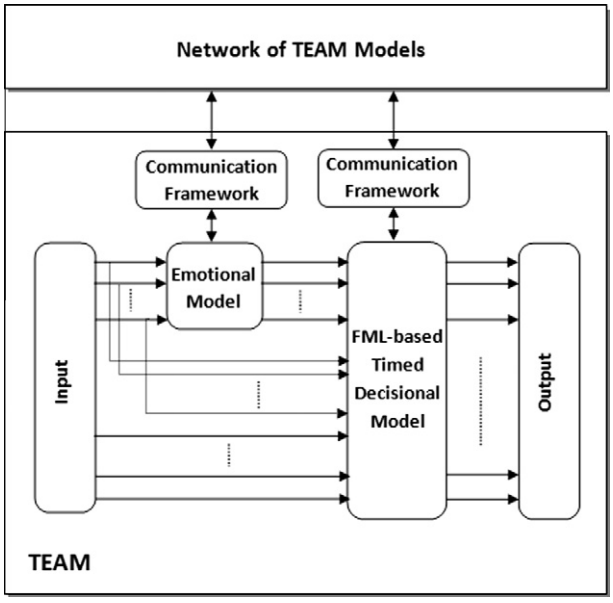

Fig. 3. TEAM architecture.

In details, the architecture of the TEAM framework (see Fig. 3) is composed of the following parts: (1) *Input Interface,* (2) *Emotional Model,* (3) *Timed Decisional Model,* (4) *Communication Framework and* (5) *Output Interface.*

The Input Interface receives environmental stimuli and forwards them to Emotional Model (EM) and Timed Decisional Model (TDM). The EM is devoted to update the emotional state of the TEAM model by using stimuli perceived from the Input Interface and by taking into account the emotional state of other TEAM models operating in the same context. The TDM utilizes input stimuli, the state of EM and its perception of time flow in order to make appropriate decisions to achieve its goal efficiently. The Communication Framework sends and receives messages, respectively, to and from intelligent entities belonging to the same collection of TEAM models. The messages contains information about: (1) emotional state of other entities and (2) decision made by other TEAM models. The Output Interface receives values computed from the Timed Decisional Model and translate them into concrete decisions and actions whose nature depends on the specific application domain.

In the following subsections, the core components of TEAM architecture represented by the EM and TDM modules are described.

4.1. Emotional module

The Emotional Module (EM) manages the emotional state of the intelligent entity as a real values m-tuple, $(i_1, i_2, \ldots, i_m)$, where each value ij, with $j = 1, \ldots, m$, represents a particular *emotional aspect* like mood and aggressiveness. In order to compute the emotional aspect values, the EM takes into account the emotion representation and the description of principal traits of human personality provided, respectively, by the OCC and OCEAN approaches. In particular, the EM manages the intelligent entity's emotional state by using the following components:

- the *emotion collection E*: it contains the set of emotions taken into account by our framework;
- a collection ξ^n of $|E|$ *intensity vectors* where the kth vector ξ^n_k contains the collection of emotion intensity computed for the kth emotion during last n time instants, i.e. at time $-n, -n-1, \ldots, -1$ where a negative time instant $-t$ indicates t instants ago;
- the *events set V*: it contains the events capable of influencing the emotional state of the intelligent entity;

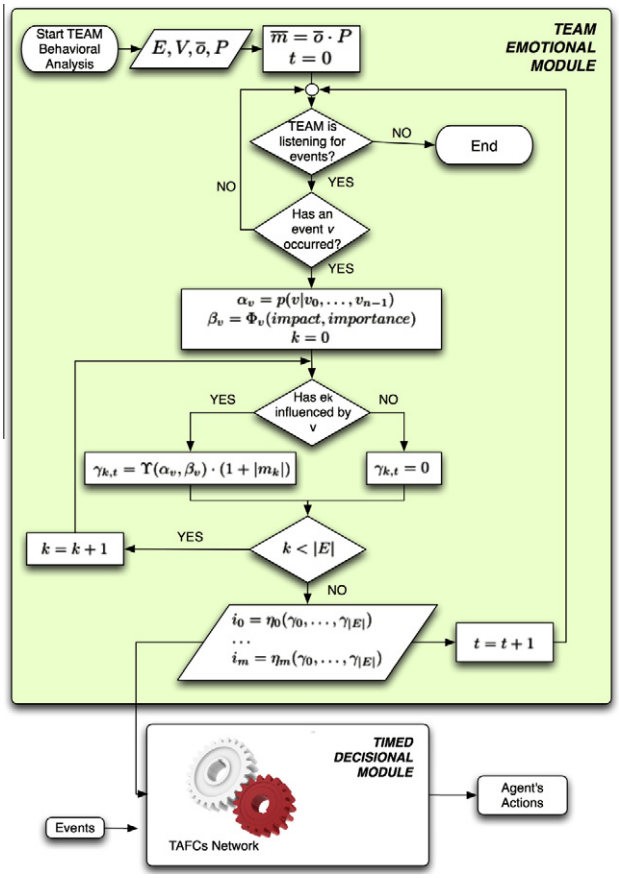

Fig. 4. TEAM workflow.

- a *row vector* $\bar{o} \in [0,1]^5$ that models the intelligent entity's personality by referring to the OCEAN model. Its values refer to the following components: *openness*, *conscientiousness*, *extraversion*, *agreeableness*, *neuroticism*;
- a *matrix* $P_{5\times|E|}$ used to specify how the different personality aspects in OCEAN influence the emotions in E. Precisely, each entry $P_{h,k}$, representing the influence of hth component of the personality on the kth emotion, is given as in [18] from:

$$P_{h,k} = \begin{cases} -0.2 & \text{if the influence is negative} \\ 0 & \text{if there is no influence} \\ 0.2 & \text{if the influence is positive} \end{cases} \quad (2)$$

- a *data structure* S, storing each occurred events sequence together with its number of occurrences.

The six-tuple $(E, \xi, V, \bar{o}, P_{5\times|E|}, S)$ enables a synergetic exploitation of OCC and OCEAN used to compute the emotional state of intelligent entity as shown in Fig. 4. More in detail, during the initialization phase, the EM module computes a row vector $\bar{m} \in [-1,1]^{|E|}$ by multiplying the $\bar{o}$ vector and the matrix $P_{5\times|E|}$. This vector is used for simulating the subjectivity whereby different intelligent entities may react to a same event. When an event v is sent by the Input Interface Module, the following steps are accomplished:

1. the *surprise degree* α_v exhibited by an intelligent entity when the event v occurs is computed as the conditioned probability $p(v|v_{n-1}, \ldots, v_0)$ of event v to occur, known that events v_{n-1}, $v_{n-2}, \ldots, v_0$ have just occurred. In order to calculate this probability the information related to occurrences of events contained in the data structure S is exploited;
2. the *desiderability* β_v related to event v is computed by a fuzzy controller Φ which takes as inputs two linguistic variables *impact* and *importance* representing, respectively, the impact of v on goals of the intelligent system and the importance of these goals. A sample of fuzzy controller rule is the following:

```
IF impact IS Very Negative AND importance IS Low
  Important
THEN desiderability_v is Low
```

1. the intensity $\gamma_{k,t}$ of each emotion $e_k \in E$ at time t (the current instant) is computed by means of the following formula:

$$\gamma_{k,t} = \Upsilon_k(\alpha_v, \beta_v) \cdot (1 + |m_k|) \quad (3)$$

where m_k is the kth element of the vector $\bar{m}$ and Υ_k is a function which determines the objective component of intensity of the kth emotion. Each emotion has a different function Υ. For example, for *happiness* the formula is:

$$\Upsilon_{happiness} = \beta_v \times \sqrt{1 - \alpha_v} \quad (4)$$

A list of functions Υ can be found in [18].

2. Each emotional aspect i_j of m-tuple $(i_1, i_2, \ldots, i_m)$ is computed by using a function η_j with two inputs: the collection of emotions $|E|$ and the set of intensity vectors ξ^n. Each emotional aspect is characterized by a different function η. For example, the formula for the emotional aspect *mood* is:

$$\eta_{mood} = \sum_{i=-n}^{-1} E_i^+ - \sum_{i=-n}^{-1} E_i^- \quad (5)$$

where E_i^+ is the intensity of positive emotions (e.g. happiness, etc.) at time i, and E_i^- is the intensity of negative emotions (e.g. sadness, etc.) at time i.

Once the collection of emotional aspects has been computed, the EM sends these values to TDM that will analyze them by a collection of TAFCs able to select the most suitable actions, as described in the next section.

4.2. Timed decisional model

The TDM module represents a key component of TEAM framework. It is a decision making module that analyzes information from the environmental context and EM and its perception of time flow to take the most appropriate sequence of actions that, simultaneously, tries to maximize both goals achievement and human likeness property of the modelled intelligent entity.

The TDM module consists of a collection of new inference engines named TAFCs. TAFCs allow a timed and qualitative evaluation of the input emotions and environmental data, by combining computational features offered by timed automata and uncertain reasoning provided by a Mamdani fuzzy system. In detail, each TAFC is devoted to manage a particular decisional aspect related to a NPC such as, for example, the managing of power-up activities (ammunitions, armors, medikits, etc.) or fight modes. Each TAFC is able to select the most suitable actions related to the corresponding decisional aspect at a given time, thanks to its dynamic features which enable it to modify itself during its life-cycle. As described in Setion 3.2, in order to define each TAFC related to a particular decision aspect two components have to be built: a fuzzy controller representing the initial control configuration and an extended timed automaton modelling the NPC's dynamic behaviour. In detail, a NPC modelled by means of a TAFC lives a sequence of discrete temporal periods named *control eras*. During each control era, the NPC's behaviour is modelled by means of a suitable fuzzy controller which represents the relative *control configuration* of the TAFC. The switching between adjacent control eras and relative

control configurations is managed through a mechanism named *control time*.

The set of TAFCs modelling different decisional aspects of a NPC run simultaneously starting from an initial time instant. As time advances, at each time unit, the TDM module performs the following steps for each designed TAFC:

- to detect the transitions which can be activated, i.e., whose condition on the clocks is satisfied or action representing an external event is occurred;
- to randomly select an active transition;
- to set to 0 all clocks present in the *clock reset* set related to the chosen transition;
- to move to the following control era;
- to update the clocks to reflect the elapsed time;
- to apply the sequence of transformation operators defined on the chosen transition. In this way, the current control configuration of the NPC related to the particular aspect at issue is changed into a new one more suitable for dealing with the new control era of NPC;
- to run the new current control configuration, i.e., the corresponding fuzzy controller, in order to produce an output for the particular decisional aspect to be sent to the Output Interface module.

After introducing TAFCs as an inference engine capable of analyzing bot emotions in a time-dependent way, FML will be introduced as the main technology for implementing the aforementioned bot intelligence in a cross-platform way.

As mentioned, the NPC behaviour is modelled by means of a set of fuzzy inference engines named TAFCs. The most suitable actions for an NPC are obtained by the inference processes executed by the current control configurations characterizing the designed TAFCs at a given time. In order to implement these control configurations in a hardware-independent way, an XML-based language named FML is used. The modelling by means of FML enhances our approach because it allows to use the designed "emotional engine" on different hardware platforms so to make it available for all video game competitors who could exploit it without additional fitting efforts.

As an example, in Fig. 5, the FML code for modelling two emotional aspects provided by EM as an input to TDM is shown.

5. A case study: exploiting TEAM to design unreal tournament 2004 bots

In order to show how the proposed TEAM framework produces an emotional, time-depending and hardware independent modeling of virtual bots, an implementation, called UnrealTEAM, of UnrealEngine Non-Player Characters (NPCs) has been used as a control system for NPCs of *Unreal Tournament 2004* (UT2k4). UT2k4 is a first-person shooter computer game used as a case study in many works [30–32]. Its principal topic is fight which represents a suitable scenario for validating our proposal. Indeed, the fight selection behaviour is strongly affected by the fighter's emotional state. In detail, during a threatening situation, primal emotions, such as fear and anger, induct people to exhibit the so-called *flight or fight* response [33,34]. Precisely, fear leads people to choose fleeing, whereas, anger result in a fight behaviour. Moreover, also more complex emotions are involved in a fight selection behaviour, e.g., satisfaction for a won fight or affliction for a lost one can influence the decision of engaging a new match [35].

The implemented framework, named UnrealTEAM, replaces UnrealEngines artificial intelligence features by using our timed emotional intelligence. In addition, the UnrealTEAM implementation is based on the integration between Pogamut and GameBots2004 that allow us to define a client/server object-oriented system to create and control our bots in an event-driven, remote and efficient way.

In particular, in this section, a bot for the game modality known as *Capture the Flag* (CTF) of UT2k4 has been realized. This bot has been called *CTFTeamBot*. The decisional module of CTFTeamBot can be designed by considering a parallel network of TAFCs whose control configurations are modelled by FML and interactions occurring in the network define bot's timed emotional behaviours. The network of TAFCs contains:

1. *Movement*: slightly adjusts the moving speed of the bot by taking into account its levels of health, burden, mood and aggressiveness and the way fatigue affects the internal perception of these levels.
2. *EngageFight*: evaluates the opportunity for the bot to engage or continue a fight against an enemy. The possible choices for the bot are: fight, escape, disengage and ignore.
3. *Fight*: instructs the bot on the modalities it has to move and shoot during a fight. Regarding the movement, possible options are charging the enemy, trying an evasive maneuver or remaining stationary. Regarding the shoot mode, instead, available options are: not shooting or shooting in primary or alternate mode. Furthermore, the bot can decide to aim its weapon or not.
4. *Power-up*: establishes when and in what measure the bot needs weapons and ammunitions, armors and medikits by considering its mood, aggressiveness and its levels of arming, armoring and health, respectively.
5. *CallHelp*: this TAFC activates itself whenever the bot starts a fight. It decides if the bot needs help from one of its fellows and in what measure, according to the psychophysical conditions of the bot, its damage potential and the value of the enemy it is facing.
6. *AnswerHelpCall*: evaluates the opportunity to run in aid of a fellow from whom has just received a help request.

In each TAFC, the current control configuration carries out the concrete decisional process by analyzing environmental events and bot emotions computed by EM. At the same time, timed automaton transitions are responsible for changing the criteria by which decisions are made by moving TAFC from a control era to the next one. At any time instant, an active subset of the listed TAFCs makes decisions required for the current situation of the bot, while the others are potentially idle. The ones which stay active depend on the sequence of events occurred and on the progression of time combined with time constraints used by the TAFCs. In any case, the decisions are always influenced by the values computed by the emotional module of the bot.

All TAFC components, i.e., control eras and control configurations, are defined by combining experts' ideas about the way to model a bot with a human appearance.

In order to better understand how UnrealTEAM framework works, the following subsections describe a portion of TDM module and how the EM Module controls an instance of CTFTeamBot during a fight scenario.

5.1. A portion of TDM module: TAFCs EngageFight and Fight

The TAFC *EngageFight* evaluates the opportunity for the bot to engage or continue a fight against an enemy. Possible choices for the bot in this situation are: *fight*, *escape* and *disengage*. As said, a TAFC is formed by an initial fuzzy controller F^0 modelled by FML and a timed control transition table T_C. The fuzzy controller F^0 is characterized by four input variables which represent the value of health (*Health*), the emotional state (*Mood*), the aggressiveness degree (*Aggressiveness*) and the value of the bot capacity to damage

```
<KnowledgeBase>
    <FuzzyVariable name=''Mood'' domainleft=''-2.0'' domainright=''2.0'' scale=''''
                   type=''input''>
        <FuzzyTerm name=''Bad'' complement=''false''>
            <TrapezoidShape Param1=''-2.0'' Param2=''-2.0'' Param3=''-1.5'' Param4=''-0.5''/>
        </FuzzyTerm>
        <FuzzyTerm name=''Neutral'' complement=''false''>
            <TrapezoidShape Param1=''-1.0'' Param2=''-0.2'' Param3=''0.3'' Param4=''1.0''/>
        </FuzzyTerm>
        <FuzzyTerm name=''Good'' complement=''false''>
            <TrapezoidShape Param1=''0.5'' Param2=''1.5'' Param3=''2.0'' Param4=''2.0''/>
        </FuzzyTerm>
    </FuzzyVariable>
    <FuzzyVariable name=''Aggressiveness'' domainleft=''-2.0'' domainright=''2.0'' scale=''''
                   type=''input''>
        <FuzzyTerm name=''Harmless'' complement=''false''>
            <TrapezoidShape Param1=''-2.0'' Param2=''-2.0'' Param3=''-1.5'' Param4=''-0.5''/>
        </FuzzyTerm>
        <FuzzyTerm name=''Combative'' complement=''false''>
            <TrapezoidShape Param1=''-1.0'' Param2=''-0.2'' Param3=''0.3'' Param4=''1.0''/>
        </FuzzyTerm>
        <FuzzyTerm name=''Aggressive'' complement=''false''>
            <TrapezoidShape Param1=''0.5'' Param2=''1.5'' Param3=''2.0'' Param4=''2.0''/>
        </FuzzyTerm>
    </FuzzyVariable>
    ....
<KnowledgeBase>
```

Fig. 5. FML code to model two emotional aspects: mood and aggressiveness.

its enemy (*Lethality*) and an output variable which represents its choice (*Decision*). Fig. 7 shows a part of the FML code modeling the fuzzy controller F^0, whereas, Fig. 6 shows the timed control transition table T_C.

T_C has three states: *MLS*, *HS* and *idle*. The *MLS* and *HS* represent two control eras in which the bot has two different needs to achieve its final aim (capturing the flag). In detail, the bot has a defined interval time to achieve its aim, so its aim becomes more important over time. Therefore, during the first control era, related to state *MLS*, the bot can be engaged in other activities, e.g, fighting, which are not its principal aim, whereas, during the control era related to state *HS*, the bot has to pay more attention to its aim rather than to other activities. So, when the bot sees an enemy or is fighting, if it is experiencing the first control era, its choices will be *fight* or *escape*, whereas, if it is in the second control era, its choice could be also *disengage*. As shown in Fig. 6, time is managed by a clock called *stc*. The initial control configuration F^0 is related to state *MLS*. In order to follow the bot strategy, fuzzy rules of fuzzy controller F^0 are characterized, as for the output variable *Decision*, by only two fuzzy terms, i.e., *fight* and *runaway*, whereas, the control configuration related to *HS* state also considers the fuzzy term *disengage*. In particular, when bot moves from the *MLS* state to the *HS* state, bot control configuration is changed by applying the operator $\boxdot^3$ on the initial configuration F^0. In details, this operator $\boxdot^3$ allows to change the consequent part of rules reg1, reg5 and reg12 (written in Fig. 7) setting the output variable *Decision* to *disengage*. The transition between states *MLS* and *HS* is activated when the clock *stc* achieves a prefixed threshold.

At the end, the state *Idle* can indicate either bot's death or end of the fight. This state is achieved starting from state *MLS* and *HS* when a particular event occurs. In detail, possible events are identified by strings *disengage?*, *seeEnemy?*, *takeDamage?* and *gameOver?* which, respectively, represent bot disengage, bot sighting of an enemy, bot damage and end of the game.

The second considered TAFC, called *Fight*, allows bot to select the fight modalities which a bot has to use during a fight against an enemy. In detail, this TAFC evaluates how the bot has to move (*MoveMode*) and how the bot has to shoot (*ShootMode*). With regard to the *MoveMode*, the choices are *pursue enemy*, *evade the enemy* and *stand still*. In addition, the bot can decide to jump (*JumpMode*). Instead, as for *ShootMode*, possible choices are *not to shoot* and use *primary* or *secondary* modality of its firearm. Apart from selecting bot shoot modalities, the TAFC evaluates the opportunity of using a manual or automatic viewfinder (*ShootAim*).

In order to model this bot behaviour, the initial fuzzy controller F^0 for the TAFC *Fight* is characterized by four output variables *MoveMode*, *JumpMode*, *ShootMode* and *ShootAim* and by three input variables which represent the distance of enemy (*EnemyDistance*),

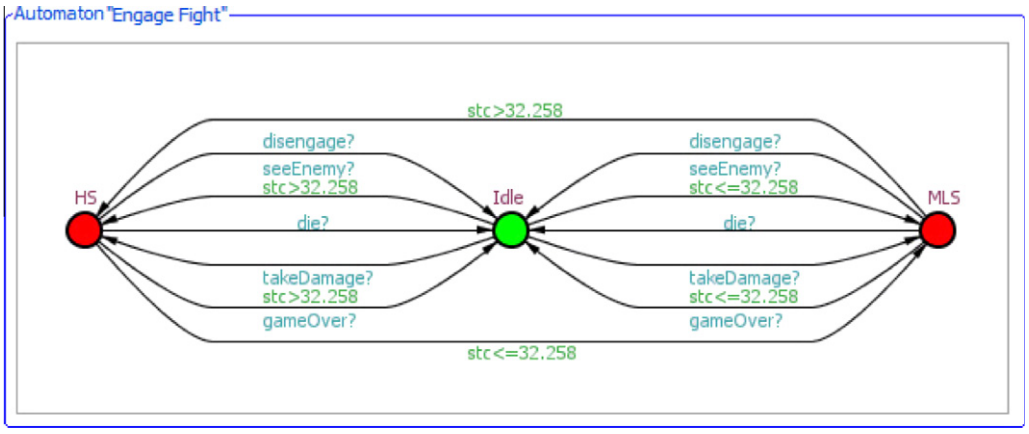

Fig. 6. Timed automaton related to the TAFC "EngageFight".

```
<FuzzyController name=''F0'' ip=''127.0.0.1''>
 <KnowledgeBase>
  <FuzzyVariable name=''Health'' domainleft=''0.0'' domainright=''100.0'' scale='''' type=''input''>
   <FuzzyTerm name=''weak'' complement=''false''>
    <TrapezoidShape Param1=''0.0'' Param2=''0.0'' Param3=''25.0'' Param4=''40.0''/>
   </FuzzyTerm>
   <FuzzyTerm name=''fine'' complement=''false''>
    <TrapezoidShape Param1=''20.0'' Param2=''40.0'' Param3=''60.0'' Param4=''80.0''/>
   </FuzzyTerm>
   <FuzzyTerm name=''strong'' complement=''false''>
    <TrapezoidShape Param1=''60.0'' Param2=''80.0'' Param3=''100.0'' Param4=''100.0''/>
   </FuzzyTerm>
  </FuzzyVariable>
  ...
  <FuzzyVariable name=''Decision'' domainleft=''1.0'' domainright=''3.0''
  scale=''null'' defaultValue=''0.0'' accumulation=''MAX'' defuzzifier=''COG'' type=''output''>
   <FuzzyTerm name=''runAway'' complement=''false''>
    <SingletonShape Param1=''1.0''/>
   </FuzzyTerm>
   <FuzzyTerm name=''disengage'' complement=''false''>
    <SingletonShape Param1=''2.0''/>
   </FuzzyTerm>
   <FuzzyTerm name=''fight'' complement=''false''>
    <SingletonShape Param1=''3.0''/>
   </FuzzyTerm>
  </FuzzyVariable>
 </KnowledgeBase>
 <RuleBase name=''RB1'' andMethod=''MIN'' orMethod=''MAX'' activationMethod=''MIN'' type=''mamdani''>
  <Rule name=''reg1'' connector=''and'' operator=''MAX'' weight=''1.0''>
   <Antecedent>
    <Clause><Variable>Mood</Variable><Term>Bad</Term></Clause>
    <Clause><Variable>Agressiveness</Variable><Term>Scared</Term></Clause>
    <Clause><Variable>Health</Variable><Term>Strong</Term></Clause>
    <Clause><Variable>Lethality</Variable><Term>Dangerous</Term></Clause>
   </Antecedent>
   <Consequent>
    <Clause><Variable>Decision</Variable><Term>fight</Term></Clause>
   </Consequent>
  </Rule>
  ......
  <Rule name=''reg5'' connector=''and'' operator=''MAX'' weight=''1.0''>
   <Antecedent>
    <Clause><Variable>Mood</Variable><Term>Bad</Term></Clause>
    <Clause><Variable>Agressiveness</Variable><Term>Combative</Term></Clause>
    <Clause modifier=''not''><Variable>Health</Variable><Term>weak</Term></Clause>
    <Clause modifier=''not''><Variable>Lethality</Variable><Term>Harmless</Term></Clause>
   </Antecedent>
   <Consequent>
    <Clause><Variable>Decision</Variable><Term>fight</Term></Clause>
   </Consequent>
  </Rule>
  .....
  <Rule name=''reg12'' connector=''and'' operator=''MAX'' weight=''1.0''>
   <Antecedent>
    <Clause><Variable>Mood</Variable><Term>Good</Term></Clause>
    <Clause><Variable>Health</Variable><Term>weak</Term></Clause>
    <Clause modifier=''not''><Variable>Lethality</Variable><Term>Harmless</Term></Clause>
   </Antecedent>
   <Consequent>
    <Clause><Variable>Decision</Variable><Term>fight</Term></Clause>
   </Consequent>
  </Rule>
  .......
 </RuleBase>
</FuzzyController>
```

Fig. 7. FML fuzzy controller F^0 for TAFC *EngageFight*.

the emotional state (*Mood*) and the aggressiveness degree (*Aggressiveness*). The fuzzy rule base of the initial fuzzy controller is defined with the aim of giving the bot a behaviour with three aspects, i.e., rationality, sensibility and causality, in order to achieve a human appearance. Fig. 8 shows a portion of the FML code modeling the fuzzy controller F^0 for TAFC *Fight*, whereas, Fig. 9 shows the timed control transition table T_C.

T_C of TAFC *Fight* has four states: *EarlyFight*, *MidFight*, *LateFight* and *Idle*. The *EarlyFight*, *MidFight*, *LateFight* represent three control eras in which the bot has been fighting for a different time length. In detail, when the bot is in the control era related to *EarlyFight* state means that has just started fighting, it is healthy and has a full control over itself, instead, during the *MidFight* control era, the bot shows signs of fatigue. Finally, during the *LateFight* control era, the bot is exhausted and it only wants the fighting to end. The time management is made through the clock called *fc*. The initial control configuration is related to state *EarlyFight*. For sake of speed, the changes applied to the initial control configuration with the aim of building the new control configurations related to states *MidFight* and *LateFight* are reported in Fig. 9. Also for TAFC *Fight*, the state *Idle* indicates either bot's death or end of the fight.

5.2. Example: a fight scenario

As an example, let us consider the emotional state of a CTF-TeamBot *B* as the 2-tuple (*Mood*, *Aggressiveness*). Let be *B* able to "feel" the following emotions:

```
<FuzzyController name=''F0'' ip=''127.0.0.1''>
  <KnowledgeBase>
    <FuzzyVariable name=''EnemyDistance'' domainleft=''0.0'' domainright=''100.0'' scale='''' type=''input''>
      <FuzzyTerm name=''far'' complement=''false''>
        <TrapezoidShape Param1=''0.0'' Param2=''0.0'' Param3=''25.0'' Param4=''40.0''/>
      </FuzzyTerm>
      <FuzzyTerm name=''medium'' complement=''false''>
        <TrapezoidShape Param1=''20.0'' Param2=''40.0'' Param3=''60.0'' Param4=''80.0''/>
      </FuzzyTerm>
      <FuzzyTerm name=''near'' complement=''false''>
        <TrapezoidShape Param1=''60.0'' Param2=''80.0'' Param3=''100.0'' Param4=''100.0''/>
      </FuzzyTerm>
    </FuzzyVariable>
    ...
    <FuzzyVariable name=''MoveMode'' domainleft=''1.0'' domainright=''3.0''
    scale=''null'' defaultValue=''0.0'' accumulation=''MAX'' defuzzifier=''COG'' type=''output''>
      <FuzzyTerm name=''Still'' complement=''false''>
        <SingletonShape Param1=''1.0''/>
      </FuzzyTerm>
      <FuzzyTerm name=''Evasive'' complement=''false''>
        <SingletonShape Param1=''2.0''/>
      </FuzzyTerm>
      <FuzzyTerm name=''Pursue'' complement=''false''>
        <SingletonShape Param1=''3.0''/>
      </FuzzyTerm>
    </FuzzyVariable>
  </KnowledgeBase>
  <RuleBase name=''RB1'' andMethod=''MIN'' orMethod=''MAX'' activationMethod=''MIN'' type=''mamdani''>
    <Rule name=''reg1'' connector=''and'' operator=''MAX'' weight=''1.0''>
      <Antecedent>
        <Clause><Variable>Mood</Variable><Term>Bad</Term></Clause>
        <Clause><Variable>Agressiveness </Variable><Term>Scared </Term></Clause>
      </Antecedent>
      <Consequent>
        <Clause><Variable>MoveMode</Variable><Term>Evade </Term></Clause>
        <Clause><Variable>ShootMode </Variable><Term>Primary </Term></Clause>
        <Clause><Variable>ShootAim </Variable><Term>None</Term></Clause>
        <Clause><Variable>JumpMode</Variable><Term>No</Term></Clause>
      </Consequent>
    </Rule>
    ......
    <Rule name=''reg5'' connector=''and'' operator=''MAX'' weight=''1.0''>
      <Antecedent>
        <Clause><Variable>Mood</Variable><Term>Bad</Term></Clause>
        <Clause><Variable>Agressiveness </Variable><Term>Aggressive </Term></Clause>
        <Clause><Variable>EnemyDistance </Variable><Term>Far </Term></Clause>
      </Antecedent>
      <Consequent>
        <Clause><Variable>MoveMode</Variable><Term>Pursue </Term></Clause>
        <Clause><Variable>ShootMode </Variable><Term>Primary </Term></Clause>
        <Clause><Variable>ShootAim </Variable><Term>None</Term></Clause>
        <Clause><Variable>JumpMode</Variable><Term>No</Term></Clause>
      </Consequent>
    </Rule>
    .....
  </RuleBase>
</FuzzyController>
```

Fig. 8. FML fuzzy controller F^0 for TAFC *Fight*.

$$E = \{Happiness, Relief, Pride, Hope, Admiration, Gratitude, Gratification, Sadness, Delusion, Shame, Fear, Reproach, Anger, Remorse\}$$

Thereafter, we refer to *Happiness* as e_0, to *Relief* as e_1 and so on.

The personality of bot B is represented by the following $\bar{o}$ vector and P matrix:

$$\bar{o} = (0.06, 0.21, 0.85, 0.61, 0.62), P = (P_+|P_-) \tag{6}$$

$$P_+ = \begin{pmatrix} 0.2 & 0 & 0.2 & 0 & 0.2 & 0.2 & 0 \\ -02 & 0 & 0.2 & 0.2 & 0.2 & 0 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0 & -0.2 & 0.2 & 0 & 0 & 0.2 & -0.2 \\ -0.2 & -0.2 & -0.2 & -0.2 & -0.2 & -0.2 & -0.2 \end{pmatrix}, P_- = -P_+ \tag{7}$$

Bot's personality influences each emotion in E according to the weights contained in the modular vector $\bar{m}$:

$$\bar{m} = \bar{o}P = \bar{o}(P_+|P_-) = \bar{o}(P_+| - P_+) = (\bar{o}P_+| - \bar{o}P_+) \tag{8}$$

$$\bar{o}P_+ = (0.016, -0.076, 0.222, 0.088, 0.100, 0.180, -0.034) \tag{9}$$

The emotional module of B keeps track of the emotions' intensity recorded in the last $n = 5$ time instants and, for any time instant $0 \leqslant t < n$, computes two emotional values $Mood_t$ and $Aggressiveness_t$ defined as follows:

$$\begin{aligned} Aggressiveness_t &= 2 \cdot \left(\max_{i=t-3}^{t} e_{11,i} - \max_{i=t-3}^{t} e_{10,i} \right) \\ Mood_t &= \sum_{k \in \{a,b\}} sign(e_k) \cdot \max_{i=t-3}^{t} e_{k,i} \end{aligned} \tag{10}$$

where

$$a = I\left(\max_{k=0}^{6} \left(\max_{i=t-3}^{t} e_{k,i}, \max_{i=t-3}^{t} e_{7+k,i} \right) \right)$$

$$b = I\left(\max_{k=0, k \neq \alpha, 7+k \neq \alpha}^{6} \left(\max_{i=t-3}^{t} e_{k,i}, \max_{i=t-3}^{t} e_{7+k,i} \right) \right)$$

$$I(e_{k,i}) = k, \quad 0 \leqslant k \leqslant 13, \ 0 \leqslant i < n$$

$$sign(e_k) = \begin{cases} 1 & \text{if } e_k \text{ is a positive emotion} \\ -1 & \text{otherwise.} \end{cases}$$

Let us consider the following game scenario: during the last five time units, bot B tried to reconquer its flag and, at the same time, during the last seven time units it was involved in a fight. Figs. 6 and 9 show that, under these circumstances, the bot TAFCs named "EngageFight" and "Fight" are, respectively, in locations named "MLS" and "LateFight". Furthermore, let us suppose that B may currently (at time 0) have a health level of 97/199, a lethality level of 49.47/100 and the following emotional situation represented by the following intensity vectors:

$$\xi_0^5 = \xi_1^5 = \xi_2^5 = \xi_3^5 = \xi_4^5 = \xi_5^5 = \xi_6^5 = [0.00\ \ 0.00\ \ 0.00\ \ 0.00\ \ 0.00]$$
$$\xi_7^5 = [0.03\ \ 0.62\ \ 0.00\ \ 0.03\ \ 0.00]$$
$$\xi_8^5 = \xi_9^5 = [0.00\ \ 0.00\ \ 0.00\ \ 0.00\ \ 0.00]$$
$$\xi_{10}^5 = [1.00\ \ 0.66\ \ 0.73\ \ 0.81\ \ 0.90]$$
$$\xi_{11}^5 = [0.03\ \ 0.66\ \ 0.73\ \ 0.03\ \ 0.03]$$
$$\xi_{12}^5 = [0.04\ \ 0.66\ \ 0.00\ \ 0.04\ \ 0.04]$$
$$\xi_{13}^5 = [0.03\ \ 0.00\ \ 0.00\ \ 0.03\ \ 0.00]$$

and, as a consequence:

$$Mood_0 = -1.73, \quad Aggressiveness_0 = -0.54 \tag{11}$$

In such a condition, the control configurations of "EngageFight" and "Fight" fire, respectively, the fuzzy rules:

```
IF Mood IS Bad AND Aggressiveness IS Combative
AND Health IS NOT Weak AND Lethality IS NOT
  Dangerous
THEN Decision IS Fight
```

```
IF Mood IS Bad AND Aggressiveness IS Combative
AND EnemyDistance IS Near
THEN MoveMode IS Evasive, ShootMode IS Primary,
  ShootAim IS None
```

As a result, the bot keeps fighting, firing its weapon in primary mode and trying to dodge enemy bullets.

If, at this point, the enemy hits the bot with a shot, the UnrealTeam module of B receives the event $v = BotDamaged \in V$ which influences only two emotions: $e_{10} = Fear$ and $e_{11} = Reproach$. If the expectation of this event is $\alpha_{BotDamaged} = 1$ and its desiderability is $\beta_{BotDamaged} = -0.4$ then the current intensities of e_{10} and e_{11} are updated as follows:

$$\begin{aligned} e_{10} &= (-\beta_{PreviousEvent} - 0.1 * \beta_{BotDamaged}) \cdot (1 + |m_{10}|) \\ &= (0.4 - 0.1 \cdot (-0.4)) \cdot (1 + |-0.088|) \approx 0.48 \end{aligned} \tag{12}$$

$$\begin{aligned} e_{11} &= (-\beta_{BotDamaged} * (1 - \alpha_{BotDamaged})) \cdot (1 + |m_{11}|) \\ &= (0.4 \cdot (1 - 1)) \cdot (1 + |-0.1|) = 0 \end{aligned} \tag{13}$$

The new emotional situation of B becomes:

$$\xi_0^5 = \xi_1^5 = \xi_2^5 = \xi_3^5 = \xi_4^5 = \xi_5^5 = \xi_6^5 = [0.00\ \ 0.00\ \ 0.00\ \ 0.00\ \ 0.00]$$
$$\xi_7^5 = [0.62\ \ 0.00\ \ 0.03\ \ 0.00\ \ 0.00]$$
$$\xi_8^5 = \xi_9^5 = [0.00\ \ 0.00\ \ 0.00\ \ 0.00\ \ 0.00]$$
$$\xi_{10}^5 = [0.66\ \ 0.73\ \ 0.81\ \ 0.90\ \ 0.48]$$
$$\xi_{11}^5 = [0.66\ \ 0.73\ \ 0.03\ \ 0.03\ \ 0.00]$$
$$\xi_{12}^5 = [0.66\ \ 0.00\ \ 0.04\ \ 0.04\ \ 0.00]$$
$$\xi_{13}^5 = [0.00\ \ 0.00\ \ 0.03\ \ 0.00\ \ 0.00]$$

and, as a consequence:

$$Mood_0 = -1.04, \quad Aggressiveness_0 = -1.94 \tag{14}$$

If, additionally, the health level of B falls down to 48.74/199, then the control configurations of "EngageFight" and "Fight" fire, respectively, the fuzzy rules:

```
IF Mood IS Bad AND Aggressiveness IS Scared AND
Health IS NOT Strong THEN Decision IS RunAway
IF Mood IS Bad AND Aggressiveness IS Scared
THEN MoveMode IS Evasive, ShootMode IS Primary,
  ShootAim IS None
```

So, the bot decides to run away, but keeps shooting and moving in an evasive manner for covering its escape.

6. Experimental results

In order to evaluate the impact of time and emotions on the human-likeness of bots and, consequently, validate the benefits provided by UnrealTEAM framework to bots behaviour modeling, a series of experiments has been conducted. In particular, in a preliminary experiment, a *CTFTeamBot* is compared with a typical bot based on an existing UT2k4 model called Hunter, which is released together with the Pogamut plug-in for NetBeans IDE. Secondly, a kind of Turing test is led in order to subjectively verify the human-likeness of our bots.

Hereafter, a detailed description of the performed experiments is reported.

6.1. Preliminary experiments

In a set of preliminary experiments our *CTFTeamBot* (referred to as T) is compared with an existing UT2k4 bot called Hunter. Hunter was born to play the Death Match combat mode of UT2k4. However, it has been easily adapted to play the CTF combat mode. The resulting bot has been called *CTFHunterBot* (referred to as H) and is driven by a logic based on IF-THEN rules and deterministic automata. The run of the experiment involves the execution of five 2-vs-2 and other five 1-vs-1 CTF matches (see Fig. 10).

The CTFTeamBot T is considered in two variants: (1) the augmented variant which is endowed with the capability of dealing with time and emotions (*with* in Table 1) and (2) the baseline variant which is not endowed with these capabilities (*without* in Table 1). By combining the number of bots in a team and the capabilities of our bot, our experiment evaluates four experimental contexts as shown in Table 1. In each match, one or two instances of our bot play against an equal number of instances of bot CTFHunterBot. In each simulation, the match is won by the team which first captures two flags. Table 1 reports scores and durations of all simulated matches. Durations are expressed in minutes and seconds. It also reports the average scores S achieved by CTFTeamBot and the average durations D expressed in seconds for each experimental context.

These preliminary experiments aim at performing an initial evaluation of human likeness level of our bot. In detail, the evaluation has been performed basing on the following assumption: the more a bot proves compelling, the more it is believable as a human being. In terms of scores and durations, this assumption translates itself into an average score close to 1 and match durations which are relatively high. Starting from this consideration, a precise evaluation of the human-likeness for each experimental context i, with $i = 1, 2, 3, 4$, is given by the following formula:

$$H_i = (1 - |1 - S_i|) + \frac{D_i}{\max\{D_1, D_2, D_3, D_4\}} \tag{15}$$

where S_i is the average of the scores computed in the ith experimental context, D_i is the average of match durations of the ith experimental context. In order to better evaluate the human likeness, a

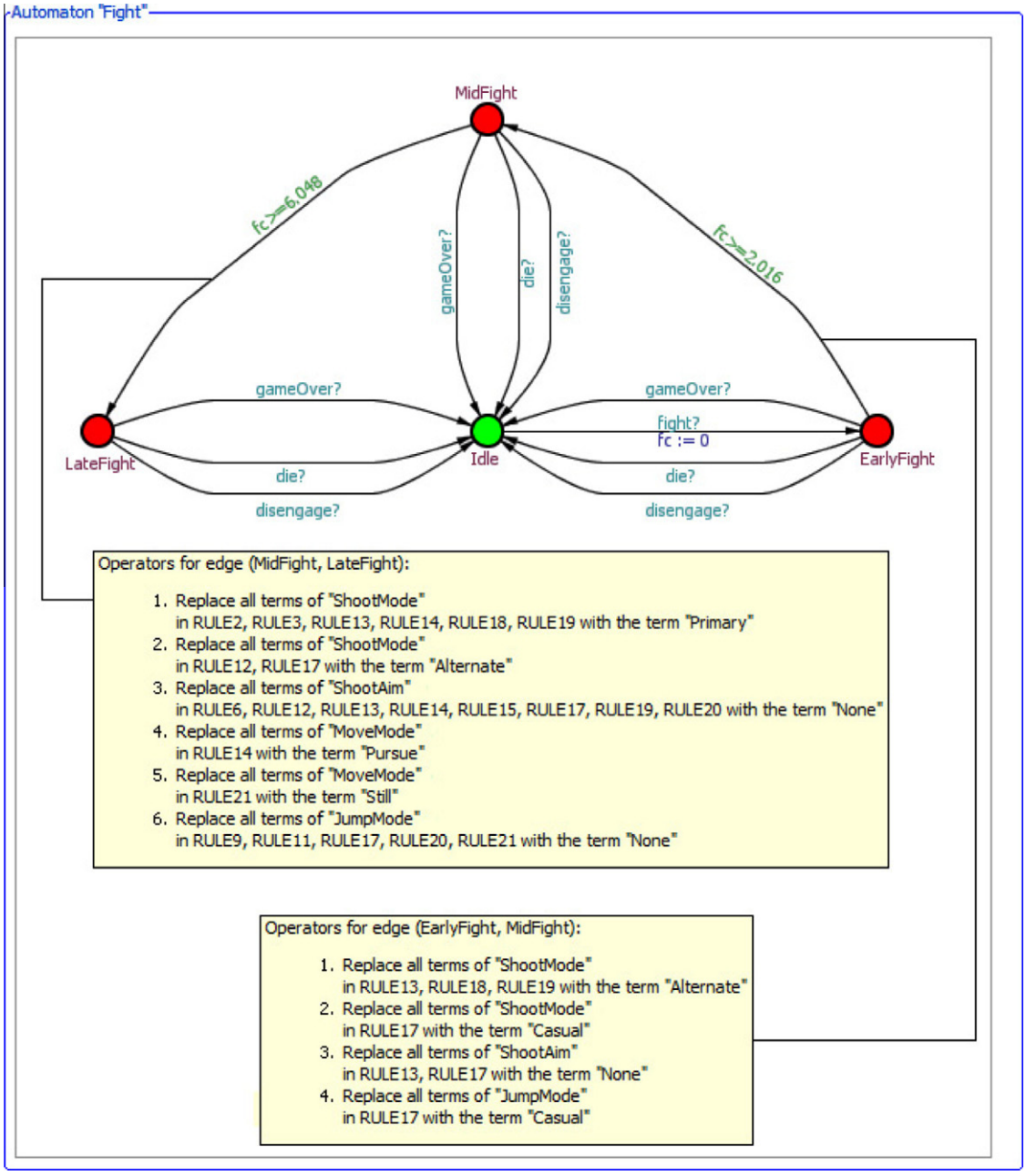

Fig. 9. Timed automaton related to the TAFC "Fight".

collection of normalized values, $\overline{H}_i$ in the interval [0, 10] is computed as follows:

$$\overline{H}_i = \frac{H_i}{\max\{H_1, H_2, H_3, H_4\}} \cdot 10.0 \tag{16}$$

where a normalized value $\overline{H}_i$ close to 10 represents a *Good human-likeness*, whereas, under 5 represents a *Bad human-likeness*. In other cases, there is an *Acceptable human-likeness*.

As shown in Fig. 1, our bot CTFTeamBot with the capability of dealing with time and emotions, used in the first and third experimental contexts, achieves the best human-likeness levels. In particular, the best performances are obtained in the third experimental context (where human-likeness level is equal to 10) since the two components, time and emotions, are fully exploited due to the interactions between team bots fighting 2-vs-2 matches.

6.2. A turing test

Since metrics score and match duration could not be complete to evaluate human-likeness level, a kind of Turing test has been performed and described in this section.

The traditional Turing test involves three participants: an interrogator, a man, and a computer. The task for the interrogator is to detect which of them is the man. Questions and answers are exchanged using a teletype machine (the equivalent of a modern chat session) [37]. If the computer is able to deceive the interrogator, it is believed to be intelligent. However, in this work, a different methodology [38] for evaluating bot's human-likeness is exploited. In detail, according to [38], during the traditional Turing test, the interrogator, representing a judge, performs a task and for this reason, he/she can devote full attention to the evaluation process.

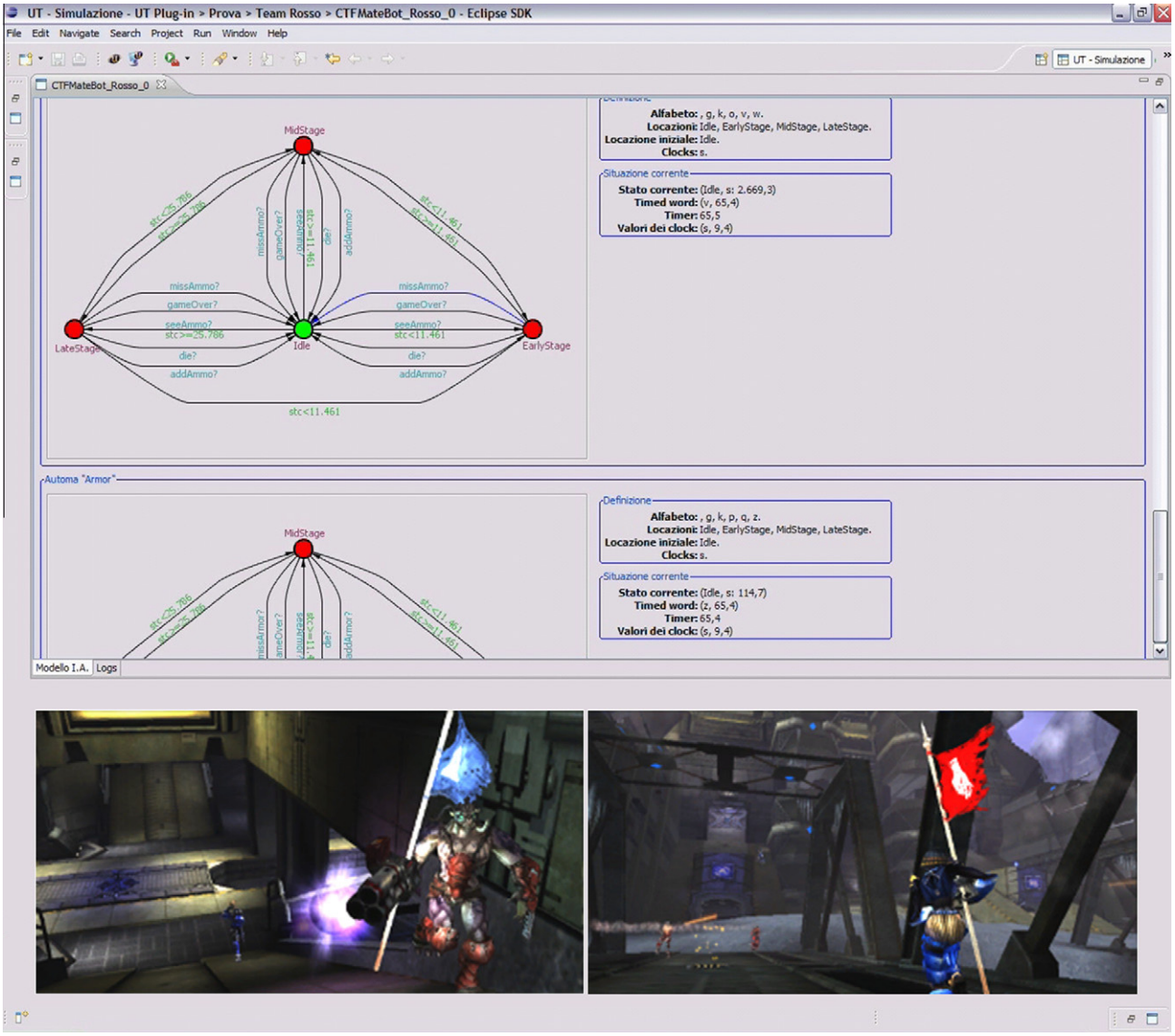

Fig. 10. A screenshot of a 1-vs-1 match between the CTFTeamBot and the CTFHunterBot.

Therefore, the new methodology does not consider the human judge to be interacting with the bot, but requires the human judge to be evaluating the bot behaviour while playing the game. By following this methodology, the Turing test was performed in three phases:

1. *Preparing videos*. In this phase, two kinds of video are recorded: in the first one, our CTFTeamBot T with the capability of dealing with time and emotions plays against a gold-standard human expert; in the second one, a human being plays against the same expert. In particular, three human beings with different capability levels (novice, medium and high) have been involved. In details, we have recorded ten videos showing a match between our CTFTeamBot and the expert, and ten videos showing a match between a human being and the expert (three for novice, three for medium and four for high).
2. *Selection of human judges*. In this phase, we have selected ten human judges among students from the University of Salerno.
3. *Performing evaluation*. In this phase, each of ten human judges has viewed four videos randomly selected (two among the first types and two among the second ones). This choice depends on the idea that watching all videos was too tiring. Each judge is asked to evaluate human-likeness on a 7-point Likert scale (values from 1 to 7).

The results of the performed Turing test show that human judges always evaluated videos recording human against expert with a human-likeness value higher than 3. With regard to videos showing our bot against expert, the human judges have evaluated them with human-likeness values also smaller than 3. However, the percentage of human evaluations higher than the number 3 is about 70%. Therefore, these results highligth that our bot shows a behaviour with a good level of human-likeness by also validating the results provided by preliminary experiments.

Table 1
Experimental results.

	Experimental context 1		Experimental context 2		Experimental context 3		Experimental context 4	
	1-vs-1				2-vs-2			
	With		Without		With		Without	
	T	*H*	*T*	*H*	*T*	*H*	*T*	*H*
1	1 03:08	2	2 01:42	0	0 04:27	2	2 03:51	0
2	0 02:32	2	2 05:42	0	0 06:33	2	2 02:31	0
3	2 05:45	1	2 02:17	0	1 05:20	2	2 04:10	0
4	0 03:34	2	2 05:00	1	1 06:34	2	2 04:20	0
5	0 04:48	2	0 05:08	2	0 03:54	2	2 02:47	0
S	0.6		1.6		0.4		2	
D	237.4 s		237.8 s		321.6 s		211.8 s	
$\overline{H}$	9.559		8.139		10.000		4.704	
Result	Good		Acceptable		Good		Bad	

7. Conclusion

In this paper a novel methodology for designing and implementing NPCs' behaviours has been introduced. Differently from other approaches, our proposal strongly considers two fundamental concepts characterizing the human being's behaviours: emotions and time. This result has been achieved by using methodologies coming from computational intelligence and psychology such as TAFCs, OCC and OCEAN. In particular, OCC and OCEAN are simultaneously used to model the personality and the collection of emotions that characterize the behaviour of a given NPC, whereas, TAFCs are used as a decision making system able to analyze video game virtual environment and NPCs' behaviour in order to select the most suitable actions improving NPCs human likeness at a given time. This hybrid approach has been developed by using FML in order to implement NPCs' behaviour in a hardware-independent way, and, consequently, achieve the so-called cross-platform timed emotional intelligence. Starting from our proposal, video games competitors will be able to both model game NPCs characterized by a human-like intelligence and move this intelligence on different platforms in a direct and simple way. As shown in the experimental results section, our approach improves game bots' human likeness and, as a consequence, enhances the video game realism.

References

[1] M. Riedl, R. Young, An objective character believability evaluation procedure for multi-agent story generation systems, in: Proceedings of the 5th International Working Conference on Intelligent Virtual Agents, 2005, pp. 278–291.

[2] P. Verduyn, I. Van Mechelen, F. Tuerlinckx, K. Meers, H. Van Coillie, Intensity profiles of emotional experience over time, Cognition Emotion 23 (7) (2009) 1427–1443.

[3] G. Acampora, F. Ferraguto, V. Loia, Synthesizing bots emotional behaviours through fuzzy cognitive processes, in: Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games, CIG2010, 2010, pp. 329–336 (art. no. 5593338).

[4] A. Ortony, G.L. Clore, A. Collins, The Cognitive Structure of Emotions, Cambridge University Press, 1988.

[5] R.R. McCrae, O.P. John, An introduction to the five-factor model and its applications, Journal of Personality (1992) 175–215.

[6] G. Acampora, V. Loia, A. Vitiello, Hybridizing fuzzy control and timed automata for modeling variable structure fuzzy systems, in: IEEE International Conference on Fuzzy Systems (FUZZ), 2010, 2010, pp. 1–8.

[7] G. Acampora, Exploiting timed automata-based fuzzy controllers and data mining to detect computer network intrusions, in: IEEE International Conference on uzzy Systems (FUZZ), 2010, 2010, pp. 1–8.

[8] G. Acampora, V. Loia, Fuzzy control interoperability and scalability for adaptive domotic framework, IEEE Transactions on Industrial Informatics 1 (2) (2005) 97–111.

[9] A. Turing, Computing machinery and intelligence, Mind 59 (236) (1950) 433–460.

[10] B. Soni, P. Hingston, Bots trained to play like a human are more fun, in: IEEE International Joint Conference on Neural Networks, 2008. IJCNN 2008 (IEEE World Congress on Computational Intelligence), 2008, pp. 363–369.

[11] P. Spronck, I. Sprinkhuizen-Kuyper, E. Postma, Online adaptation of computer game opponent AI, in: Proceedings of the 15th Belgium-Netherlands Conference on Artificial Intelligence, 2003, pp. 291–298.

[12] B. Gorman, C. Thurau, C. Bauckhage, M. Humphrys, Believability testing and Bayesian imitation in interactive computer games, in: Proc. 9th Int. Conf. on the Simulation of Adaptive behaviour (SAB06), LNAI, Springer, 2006.

[13] Richard Zhao, Duane Szafron, Generating believable virtual characters using behaviour capture and Hidden Markov models, in: Advances in Computer Games 13 Conference, 2011.

[14] C.D. Elliott, The Affective reasoner: a process model of emotions in a multi-agent system, Unpublished Ph.D. thesis, The Institute for the Learning Sciences, Northwestern University, Evanston, Illinois, 1992.

[15] W.S.N. O'Reilly, Believable social and emotional agents, Unpublished Ph.D. Thesis, Carnegie Mellon University, Pittsburgh, PA, 1996.

[16] C. Bartneck, Integrating the occ model of emotions in embodied characters, in: Proceedings of the Workshop on Virtual Conversational Characters, 2002, p. 3948.

[17] D. Freeman, Creating emotion in games: the craft and art of emotioneering, Comput. Entertain. 2 (3) (2004). 15–15.

[18] D.A. Tvara, A. Meier, Agents with personality for videogames, in: IV Conference of Articulated Motion and Deformable Objects (AMDO 2006), Palma de Mallorca, Spain, 2006, pp. 484–493.

[19] C. Delgado-Mata, J. Ibez-Martnez, F. Gmez-Caballero, O. Guilln-Hernndez, Behavioural reactive agents to define personality traits in the videogame berpong, in: Z. Pan, A. Cheok, W. Mller, A. El Rhalibi (Eds.), Transactions on Edutainment I, Lecture Notes in Computer Science, vol. 5080, Springer, Berlin/Heidelberg, 2008, pp. 135–149.

[20] T.B. Yoon, D.M. Kim, K.H. Park, J.H. Lee, K.-H. You, Game player modeling using D-FSMs, in: Michael J. Smith, Gavriel Salvendy (Eds.), Proceedings of the 2007 Conference on Human Interface: Part II, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 490–499.

[21] D. Hirono, R. Thawonmas, Implementation of a human-like bot in a first person shooter: second place bot at botprize 2008, in: Proc. Asia Simulation Conference 2009 (JSST 2009), Ritsumeikan University, Shiga, Japan, 2009, p. 5.

[22] I.J. Roseman, A.A. Antoniou, P.E. Jose, Appraisal determinants of emotions: constructing a more accurate and comprehensive theory, Cognition Emotion 10 (1996) 241–277.

[23] A. Sloman, Architectural requirements for human-like agents both natural and artificial, in: K. Dautenhahn (Ed.), Human Cognition And Social Agent Technology, Advances in Consciousness Research, John Benjamins Publishing Company, Amsterdam, 1999.

[24] C. Adam, A. Herzig, D. Longin, A logical formalization of the OCC theory of emotions, Synthese 168 (2) (2009) 201–248.

[25] G. Ruebenstrunk, Emotional Computers: Computer models of emotions and their meaning for emotion-phychological research, 1998. <http://www.ruebenstrunk.de/emeocomp/content.HTM>.
[26] T.-H.-H. Dang, S. Letellier-Zarshenas, D. Duhaut, Comparison of recent architectures of emotions, in: The 10th International Conference on Control, Automation, Robotics and Vision ICARCV 2008, Hanoi, Vietnam, 2008.
[27] M. Seif El-Nasr, J. Yen, T.R. Ioerger, FLAME – fuzzy logic adaptive model of emotions, Autonomous Agents and Multi-Agent Systems, Vol. 3, Kluwer Academic Publishers, Netherlands, 2000, pp. 219–257.
[28] T.D. Bui, D. Heylen, M. Poel, A. Nijholt, ParleE: an adaptive plan based event appraisal model of emotions, in: Parlevink Internal Report, University of Twente, 2002.
[29] R. Alur, A theory of timed automata, Theor. Comput. Sci. 126 (1994) 183–235.
[30] D. Trenholme, S. Smith, Computer game engines for developing first-person virtual environments, in: Virt. Real., Springer, London, 2008.
[31] R.L. Hy, A. Arrigoni, P. Bessiere, O. Lebetel, Teaching Bayesian behaviours to video game characters, Robot. Auton. Syst. 47 (2004) 177–185.
[32] D. Wang, B. Subagdja, A. Tan, G. Ng, Creating human-like autonomous players in real-time first person shooter computer games, in: Proc. IAAAI -09, 2009.
[33] W.B. Cannon, Bodily Changes in Pain, Hunger, Fear, and Rage, Appleton-Century-Crofts, New York, 1929.
[34] F.G. Graeff, Neuroanatomy and neurotransmitter regulation of defensive behaviours and related emotions in mammals, Braz. J. Med. Biol. Res. 27 (1994) 81129.
[35] G. Zhang, Z. Li, The emotion mechanism of artificial life fight behaviour, in: International Conference on Computational Intelligence and Natural Computing. in CINC '09, vol. 1, 2009, pp. 15–18.
[36] G. Acampora, V. Loia, An open integrated environment for transparent fuzzy agents design, in: Open Source Development, Communities and Quality, IFIP International Federation for Information Processing, vol. 275/2008, Springer, Boston, 2008, pp. 1571–5736.
[37] P. Hingston, A turing test for computer game bots, IEEE Transactions on in Computational Intelligence and AI in Games 1 (3) (2009) 169–186.
[38] J.E. Laird, J.C. Duchi, Creating human-like synthetic characters with multiple skill levels: a case study using the soar quakebot, in: 2001 AAAI Spring Symposium on Artificial Intelligence and Computer Games, 2001, p. 5458.