Modeling and Implementing an Emotional Based Decision Agent

Pablo G. Esteban
Department of Statistics and Operations Research
Universidad Rey Juan Carlos
Madrid
pablo.gomez.esteban@urjc.com

Abstract

We provide a model that supports the decision making process of an autonomous agent which interacts with several users and is influenced by affective mechanisms. The approach has a decision analytic flavor, but includes models forecasting the users' behavior. Moreover, simulated emotions impact the agent's utility function. We describe the implementation of the model with an edutainment bot.

Keywords: Decision Analysis, Adversarial Risk Analysis, Affective Decision Making, Affective Computing, Robotics

1 Introduction

Recently, the field of cognitive processes has shown that emotions may have a direct impact on decision-making processes, see e.g. [28]. Advances in areas such as affective decision making [5], neuroeconomics [7] and affective computing [17] are based on this principle.

Following this, we would like to provide a model for an autonomous agent that makes decisions influenced by emotional factors when interacting with humans and other agents. For that purpose, we have based our work on the recently introduced framework of Adversarial Risk Analysis (ARA) [19], which avoids the standard and unrealistic game theoretic assumptions of

common knowledge, through a nested hierarchy of decision analysis models. From the point of view of our agent, the problem is understood as a decision analytic one, but we consider principled procedures which employ the adversarial structure to forecast the adversaries' (other agents or humans) actions. On doing this, the agent should forecast what the other participants think about him, thus starting the above mentioned hierarchy. Depending on the level the agent climbs up in such hierarchy, we would talk about 0-level analysis, 1-level analysis and so on, borrowing the k-level thinking terminology and view, see [22], [1] and [11]. Our approach has a clear Bayesian game theoretic flavor, as in [12] and [18].

Our model is essentially decision analytic, see [4], incorporating also models forecasting the evolution of its adversaries and the environment surrounding all of them. We also include models simulating emotions, which have an impact on the agent's utility function. We aim at supporting the decision making of an emotional based agent improving interfacing and interaction with users. We describe the implementation of our model with a edutainment bot endowed with several sensors to infer users' actions and environment's states.

The structure of the paper is as follows. In Section 2, we describe the basic elements and participants in our framework. Section 3, defines the incumbent forecasting and preference models and the expected utility maximization based on some dynamic programming techniques. The effective implementation of our model is in Section 4, including a detailed explanation of forecasting and multiobjective preference models. The paper ends with a discussion and future work on Section 5.

2 Basic framework

We start by introducing the basic elements of our model. We aim at designing an agent A whose activities we want to regulate and plan. There are participants or users, $B_1, \ldots, B_r \in \mathcal{B}_u$, which interact with A. An index $u \in \{1, 2, ..., r\}$ will identify the corresponding user. The activities of both A and the $B_u{}'s$ take place within an environment E. As a motivating example, suppose that we aim at designing a bot A which will interact with a group of three kids, B_1, B_2, B_3, within a room E.

A makes decisions within a finite set $\mathcal{A} = \{a_1, \ldots, a_m\}$, which possibly includes a *do nothing* action. The $B_u{}'s$ make decisions within a set $\mathcal{B} =$

$\{b_1, \ldots, b_n\}$, which also includes a *do nothing* action. $\mathcal{B}$ will be as complete as possible, while simplifying all feasible alternatives down to a finite number. The environment E changes with the users' actions, adopting states within a set $\mathcal{E}$.

The agent faces this changing environment, which affects its own behavior. A has q sensors providing readings about the external environment. Each sensor reading is attached to a time t, so that the sensor reading vector is $s_t = (s_t^1, \ldots, s_t^q)$. The agent infers the external environmental state e, based on a possibly probabilistic transformation function f, so that

$$\hat{e}_t = f(s_t).$$

A also uses the sensor readings to infer which user he is facing, through a probabilistic function h

$$\hat{B}_t = h(s_t).$$

Finally, A employs the sensor readings to infer what the users have done, based on a (possibly probabilistic) function g

$$\hat{b}_t = g(s_t).$$

We design our agent by planning its activities according to the basic loop in Fig. 1, which is open to interventions, see [26], if an exception occurs.

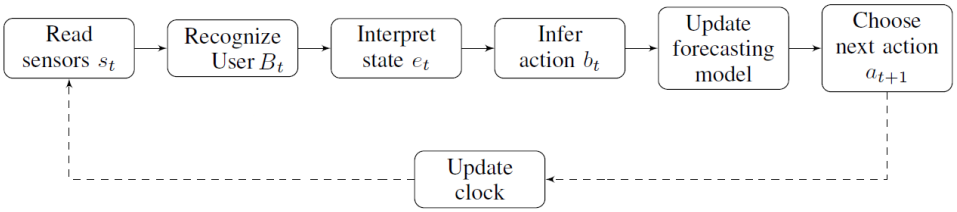

Figure 1: Basic Agent Loop

As we can see in Fig. 1, at the begining of each iteration, sensors s_t are read in order to recognize the identifiable users $\mathcal{B}_u$, as well as interpret the state e_t and infer the users' actions b_t. Once the bot knows about what is around him, the next step is to update the forecasting model with the information he just received through sensors s_t. Using the recently updated forecasting model, he maximizes the expected utility of his possible actions to choose the next one. The loop is completed by updating the clock.

3 ARA affective decision model

Essentially, we shall plan our agent's activities over time within the decision analytic framework, see [4]. We describe, in turn, the forecasting model, which incorporates ARA elements, the preference model and emotions, and, finally, the corresponding optimization problem.

3.1 Forecasting models

The agent maintains a forecasting model which suggests with which probabilities will the users act and the environment react, given the past history of the agent's actions, the users' actions and the evolution of the environment $(e_{t-1}, a_{t-1}, b_{t-1})$ and its action a_t.

We describe the general structure of our models. Assume that, for computational reasons, we limit the agent's memory to two instant times. For the moment, we shall just forecast one period ahead. Therefore, we are interested in computing, for each user B_u,

$$p(e_t, b_t \mid a_t, (e_{t-1}, a_{t-1}, b_{t-1}), (e_{t-2}, a_{t-2}, b_{t-2}), B_u). \tag{1}$$

(1) may be decomposed through

$$p(e_t|b_t, a_t, (e_{t-1}, a_{t-1}, b_{t-1}), (e_{t-2}, a_{t-2}, b_{t-2}), B_u) \times$$
$$\times\ p(b_t|a_t, (e_{t-1}, a_{t-1}, b_{t-1}), (e_{t-2}, a_{t-2}, b_{t-2}), B_u).$$

We refer to the first term which we call the *environment model*. We assume that the environment is fully under control by the users. In our motivating example, they control the light, the temperature and other features of the room. Moreover, they may plug in the bot to charge its battery, and so on. Only the latest of the users' actions will trigger the evolution of the environment. Thus, we shall assume that, in general,

$$p(e_t \mid b_t, a_t, (e_{t-1}, a_{t-1}, b_{t-1}), (e_{t-2}, a_{t-2}, b_{t-2}), B_u) = p(e_t \mid b_t, e_{t-1}, e_{t-2}).$$

Regarding the second term, we shall consider that the users have their own behavior evolution, that might be affected by how they react to the agent's actions, thus incorporating the ARA principle. Thus, we assume that

$$p(b_t \mid a_t, (e_{t-1}, a_{t-1}, b_{t-1}), (e_{t-2}, a_{t-2}, b_{t-2}), B_u) = p(b_t \mid a_t, b_{t-1}, b_{t-2}, B_u). \tag{2}$$

The agent will maintain two models in connection with (2) for each user. The first one, model M1, describes the evolution of the users by themselves, assuming that they are in control of the whole environment, and they are not affected by the agent's actions. We call them the *users' models*, described through

$$p(b_t \mid b_{t-1}, b_{t-2}, B_u).$$

The other one, model M2, refers to the users' reactions to the agent's actions. Indeed, it assumes that the users are fully reactive to the agent, which we describe through

$$p(b_t \mid a_t, B_u).$$

We call them the *classical conditioning models*, with the agent possibly conditioning the users. We combine both models to recover (2). We view the problem as one of model averaging, see [10]. In such case,

$$p(b_t \mid a_t, b_{t-1}, b_{t-2}, B_u) =$$

$$= \Big[p(M_2 \mid B_u)\ p(b_t \mid b_{t-1}, b_{t-2}, B_u) + p(M_1 \mid B_u)\ p(b_t \mid a_t, B_u) \Big],$$

where $p(M_i \mid B_u)$ denotes the probability that the agent gives to model Mi, given that the user is B_u, with $p(M_1 \mid B_u) + p(M_2 \mid B_u) = 1,\ p(M_i \mid B_u) \geq 0$. These probabilities, essentially, capture how reactive to the agent's actions the users are.

Finally, based on the above, we shall use

$$p(e_t, b_t \mid a_t, (e_{t-1}, a_{t-1}, b_{t-1}), (e_{t-2}, a_{t-2}, b_{t-2})) =$$

$$= \sum_u \Big[p(e_t \mid b_t, e_{t-1}, e_{t-2}) \times p(b_t \mid a_t, b_{t-1}, b_{t-2}, B_u) \times p(B_u) \Big].$$

Extensions to forecasting m steps ahead follow a similar path. For example, for two steps ahead, we have

$$p((e_{t+1}, b_{t+1}), (e_t, b_t) \mid a_t, (e_{t-1}, a_{t-1}, b_{t-1}), (e_{t-2}, a_{t-2}, b_{t-2})) =$$

$$= p((e_{t+1}, b_{t+1}) \mid a_t, (e_{t-1}, a_{t-1}, b_{t-1}), (e_{t-2}, a_{t-2}, b_{t-2}), (e_t, b_t)) \times$$

$$\times\ p(e_t, b_t \mid a_t, (e_{t-1}, a_{t-1}, b_{t-1}), (e_{t-2}, a_{t-2}, b_{t-2})) =$$

$$= \sum_u \left[p(e_{t+1} \mid b_{t+1}, b_t, e_t, e_{t-1}, e_{t-2}) \times p(b_{t+1} \mid a_t, b_t, b_{t-1}, b_{t-2}, B_u^{t+1}) \times p(B_u^{t+1}) \right] \times$$

$$\times \sum_u \left[p(e_t \mid b_t, e_{t-1}, e_{t-2}) \times p(b_t \mid a_t, b_{t-1}, b_{t-2}, B_u^t) \times p(B_u^t) \right].$$

Learning about various models within our implementation is sketched in Section 4.

3.2 Affective preference model

We describe now the preference model. Assume that the agent faces consequences $c = (c_1, c_2, \ldots, c_l)$. At each instant t, they will depend on his action a_t, the users' action b_t and the future state e_t, realized after a_t and b_t. Therefore, the consequences will be of the form

$$c_i(a_t, b_t, e_t),\ i = 1, \ldots, l.$$

We assume that they are evaluated through a multi-attribute utility function, see [4]. Specifically, without much loss of generality, as argued in [27], we shall adopt an additive form

$$u(c_1, c_2, \ldots, c_l) = \sum_{i=1}^{l} w_i u_i(c_i),$$

with $w_i \geq 0,\ \sum_{i=1}^{l} w_i = 1$.

The consequences might be perceived differently depending on the current emotional state $d_t(B_u)$ that the agent has, assuming that the identified user is B_u. Indeed, emotions might change depending on the user the agent is facing. We shall define such emotional state in terms of the level of k basic emotions, through a mixing function

$$d_t = h(em_t^1, em_t^2, \ldots, em_t^k).$$

[6] and [23] provide many pointers to the literature on mixing emotions. The intensity of these basic emotions, in turn, will be defined in terms of how desirable a situation is, i.e., how much utility $u(c_t)$ is gained, and how surprising the situation was, see [8] for an assessment of various models in relation with emotions. The expectations, or surprise, will be defined by comparing the predicted and the inferred users' actions through some distance function

$$z_t = \delta(\bar{b}_t, \hat{b}_t),$$

where $\bar{b}^t$ is (the most likely) predicted action for user B_u.

We assume some stability within emotions, in that current emotions influence future emotions. Thus, we consider a probabilistic evolution of emotions through

$$em_t^i = r_i(em_{t-1}^i, u(c_t), z_t),$$

as in [28]. Finally, following [14], we shall actually consider that the utility weights will depend on the emotional state, the stock of visceral factors in their notation, so that, at time t

$$u(c) = \sum_{i=1}^{l} w_i(d_t(B_u))u_i(c_i).$$

3.3 Expected utility

The goal of our agent will be to maximize the predictive expected utility. Planning several instants ahead requires computing maximum expected utility plans defined through:

$$\max_{(a_t,\ldots,a_{t+r})} \psi(a_t,\ldots,a_{t+r}) = \sum_{(b_t,e_t),\ldots,(b_{t+r},e_{t+r})} \left[\sum_{i=D}^{r} u(a_{t+i}, b_{t+i}, e_{t+i})\right] \times$$

$$\times\, p((b_t,e_t),\ldots,(b_{t+r},e_{t+r}) \mid (a_t, a_{t+1},\ldots,a_{t+r},(a_{t-1},b_{t-1},e_{t-1}),(a_{t-2},b_{t-2},e_{t-2}))).$$

assuming utilities to be additive over time. This could be solved through dynamic programming, the Bellman's equation [2] being in this case

$$V_t((e_{t-2},b_{t-2}),(e_{t-1},b_{t-1})) = \max_{a_t} \int [u(a_t,b_t,e_t) + V_{t+1}((e_{t-1},b_{t-1}),(e_t,b_t))] \times$$

$$\times\, p(e_t|b_t,e_{t-1},e_{t-2})\; p(b_t|a_t,b_{t-1},b_{t-2}) db_t de_t$$

If planning several instants ahead turns out to be very expensive computationally, we could plan just one period ahead. In this case, we would aim at solving

$$\max_{a_t \in \mathcal{A}} \psi(a_t) = \sum_{b_t, e_t} u(a_t, b_t, e_t) \times$$
$$\times \ [p(b_t, e_t \mid a_t, (a_{t-1}, b_{t-1}, e_{t-1}), (a_{t-2}, b_{t-2}, e_{t-2}))] \, .$$

We may mitigate the myopia of this approach by adding a term penalizing deviations from some ideal agent consequences, as in [21]. In this case, the utility would have the form $u(c) - \rho(c, c^*)$ where ρ is a distance and c^* is an ideal consequence value.

Agents operating in this way may end up being too predictable. We may mitigate such effect by choosing the next action in a randomized way, with probabilities proportional to predictive expected utilities, that is

$$P(a_t) \propto \psi(a_t),$$

where $P(a_t)$ is the probability of choosing a_t. See [16] for a justification of such approach.

4 Implementation

The above procedures have been implemented within the AISoy1 bot environment (http://www.aisoy.es). Some of the details of the model implemented are described next, with code developed in C++ over Linux. This bot has several sensors including a camera to detect objects or persons within a scene; a microphone used to recognize when the user talks and understand what they say, through an ASR component; several touch sensors to interpret when it has been stroked or attacked; an inclination sensor so as to know whether it is in vertical position or not; a light sensor and a temperature sensor. As actuators, it includes some servoes that allow it to move some parts of its body to express emotions, but it mostly uses a text-to-speech system (TTS) combined with a led matrix to simulate its mouth when talking or expressing happiness or sadness. Its central led light is used to show the emotion is experiencing at an specific moment. The information provided by these sensors is used by the bot to detect the user and infer the users' actions and the environmental states.

4.1 Basic elements

The bot's alternatives in $\mathcal{A}$ include actions for complaining, some ways of calling the users' attention, several options to interact with the users and a *do nothing* action, as described in Fig. 2. This totals $m = 15$ alternatives for the bot, with $\mathcal{A} = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, a_{14}, a_{15}\} =$ {*cry, alert, warn, ask for help, salute, play, speak, ask for playing, ask for charging, ask for shutting down, tell jokes, tell stories, tell events, obey, do nothing*}. For instance, the *ask for help* action consists of calling for an identified user when the bot detects an unknown user within the scene or whether it is feeling fear for some reason. Complain actions are ordered according to their intensity: the *warn* action would be triggered when the bot inferes some disgusting users' actions; if the users keep on doing the same action, the bot will use the *alert* action; and so on.

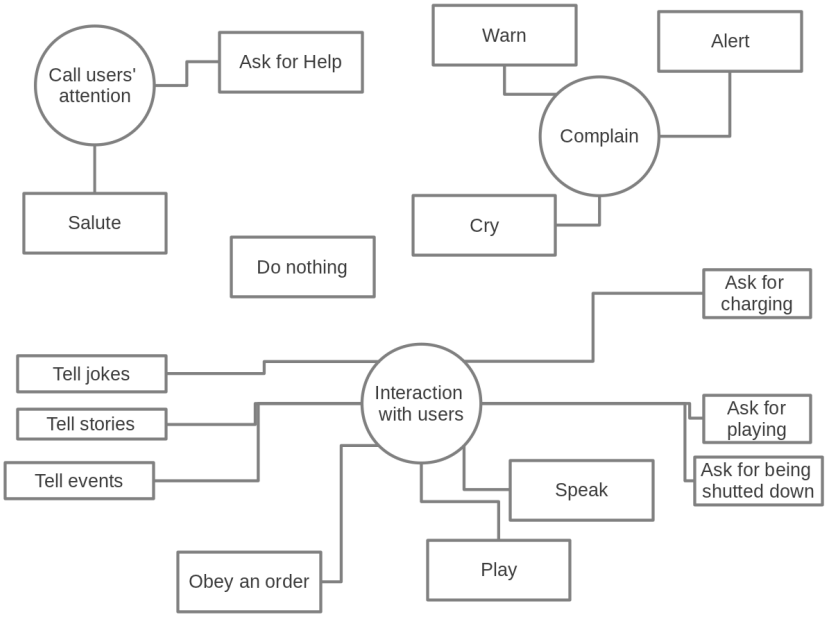

Figure 2: Bot actions

On the users' side, set $\mathcal{B}$, the bot is able to detect several users' actions, some of them in a probabilistic way. Indeed, the bot detects three types of actions: affective, aggressive, and interacting actions, see Fig. 3. The bot will also detect whether none of the users made *no action*. This totals $n =$ 12 actions with $\mathcal{B} = \{b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8, b_9, b_{10}, b_{11}, b_{12}\} = \{$ *recharge,*

stroke, flatter, attack, offend, move, update, speak, play, order, ignore, do nothing }. As mentioned, we shall assume that this set is fixed, but we shall outline in the discussion how it could be increased.

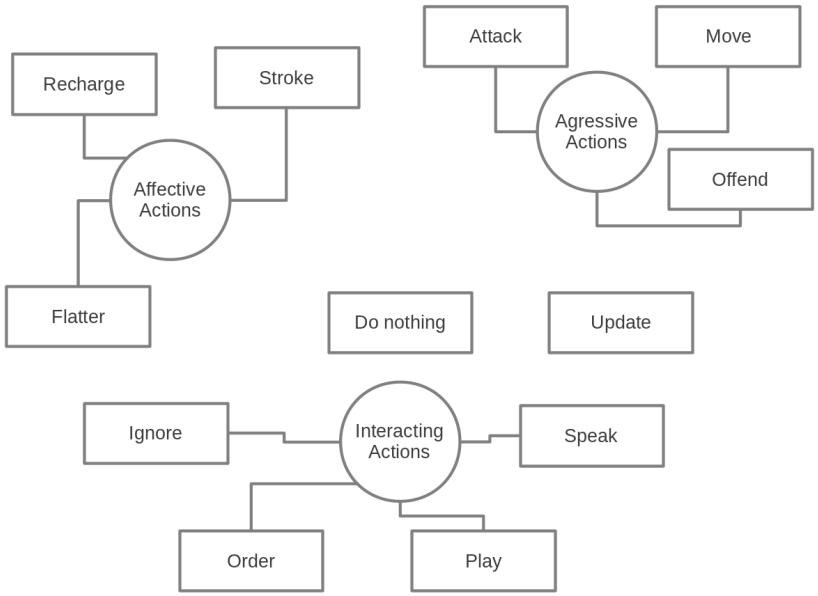

Figure 3: Users' actions

The detection of some actions is based on simple deterministic rules. For example, the attack action is interpreted through a detection in a touch sensor and a variation in the inclination sensor. Others are detected according to probabilistic rules, like those involving voice recognition and processing. We provide a sketch of how the users' actions are detected:

- $b_1 : recharge$. Rule: Battery Charge < 100 % and power supply cable connected and power supply status different from last power supply status.
- $b_2 : stroke$. Rule: While it is in vertical position, there is no change in inclination at the next *2 instants* and is touched during the following *2 instants*.
- $b_3 : flatter$. Rule: Detection of words within a specific set [rewards, compliments, etc.] and detects the presence of the user or the name of the bot.

- b_4 : *attack*. Rule: There are changes in the inclination at the following *2 instants* or the bot is not in vertical position.
- b_5 : *move*. Rule: There are changes in the inclination at the following *2 instants* and contact detected during the following *2 instants*.
- b_6 : $offend$. Rule: Detection of words in a specific set [insults, threats, etc.] and detects the presence of the user or the name of the bot.
- b_7 : *ignored*. Rule: Detects the presence of the user and there is no response from him at the following *2 instants*.
- b_8 : *speak*. Rule: Detects the presence of the user or the name of the bot and the user starts an speaking grammar set.
- b_9 : *play*. Rule: Detects the presence of the user or the name of the bot and the user asks for playing.
- b_{10} : *order*. Rule: Detects the presence of the user or the name of the bot and the user asks for an action within a set.
- b_{11} : *do nothing*. Rule: Detects the presence of the user and the user does not do any of the defined actions and the bot is in vertical position.
- b_{12} : *update*. Rule: Detects a difference in the software version when the bot is rebooted.

Regarding the environment, the bot may recognize, through its sensors, contextual issues such as the presence of noise or music, the level of darkness, the temperature, or its inclination, as described below.

4.2 Forecasting model

We describe now how we have implemented the relevant forecasting models. D_t will designate the data available until time t.

4.2.1 The classical conditioning model

This model forecasts the users' actions based on the agent action for each B_u. We shall use a matrix-beta model for such purpose [20]. For each a_t, the prior distribution will be Dirichlet, so that

$$p(b_t \mid a_t = a_j, B_u) \sim Dir(\beta^u_{1j}, \ldots, \beta^u_{nj}),\ b_t \in \{b_1, b_2, ..., b_n\}.$$

Now, if h^u_{ij} designates the number of occurrences of user B_u doing b_i, when the bot has made a_j, the posterior distribution will be

$$p(b_t \mid a_t = a_j, D_t, B_u) \sim Dir(\beta^u_{1j} + h^u_{1j}, \ldots, \beta^u_{nj} + h^u_{nj}),\ b_t \in \{b_1, b_2, ..., b_n\}.$$

When necessary, we may summarize it through its average

$$\hat{p^u_{ij}} = \frac{\beta^u_{ij} + h^u_{ij}}{\sum_i (\beta^u_{ij} + h^u_{ij})},\ i \in \{1, 2, ..., n\}, j \in \{1, 2, ..., m\}.$$

The required data will be stored in the matrix structure,

$$\begin{pmatrix} \beta^t_{11} = \beta_{11} + h_{11} & \cdots & \beta^t_{1m} = \beta_{1m} + h_{1m} \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ \beta^t_{n1} = \beta_{n1} + h_{n1} & \cdots & \beta^t_{nm} = \beta_{nm} + h_{nm} \\ \beta^t_{(n+1)1} = \sum_{i=1}^{n} (\beta_{i1} + h_{i1}) & \cdots & \beta^t_{(n+1)m} = \sum_{i=1}^{n} (\beta_{im} + h_{im}) \end{pmatrix}$$

whose last row accumulates the sum of row values for each column. There will be one of these structures for each user. At each time instant, we shall increment the corresponding ij-th element of the matrix and the corresponding element of the last row: if the sequence is a_j, b_i, we shall update $\beta^{u(t+1)}_{ij} = \beta^{ut}_{ij} + 1$ and $\beta^{u(t+1)}_{(n+1)j} = \beta^{ut}_{(n+1)j} + 1$, with the rest of entries satisfying $\beta^{u(t+1)}_{ij} = \beta^{ut}_{ij}$.

Since we expect lots of data, the terms β^u_{ij} will not matter that much after a while. Thus, we shall use the following prior assessment: if a pair of actions $a_t = a_j$ and $b_t = b_i$ are compatible, we shall make $\beta^u_{ij} = 1$; otherwise, we shall make $\beta^u_{ij} = 0$.

4.2.2 The users' model

We provide now our forecasting model for the current users' action based on what the users have done two time steps before. As before, we use a matrix-beta model for each user, assuming that b_{t-1} and b_{t-2} are associated with the same user. For $i, j \in \{1, 2, \ldots, n\}$, we have a priori

$$p(b_t \mid b_{t-1} = b_i, b_{t-2} = b_j, B_u) \sim Dir(\beta^u_{1ij}, \ldots, \beta^u_{nij}),\ b_t \in \{b_1, b_2, ..., b_n\}.$$

If h^u_{kij} designates the number of occurrences that the user B_u did b_k after having done b_i and b_j, we have that the posterior is

$$p(b_t \mid b_{t-1} = i, b_{t-2} = j, D_t, B_u) \sim Dir(\beta^u_{1ij}+h^u_{1ij}, \ldots, \beta^u_{nij}+h^u_{nij}),\ b_t \in \{b_1, b_2, ..., b_n\},$$

which we may summarize, when needed, through

$$\hat{p}_{kij} = \frac{\beta^u_{kij} + h^u_{kij}}{\sum_k (\beta^u_{kij} + h^u_{kij})},\ k \in \{1, 2, ..., n\}.$$

The data structure used to store the required information will consist of a three-dimensional matrix as in Fig. 4, and there will be one for each user B_u.

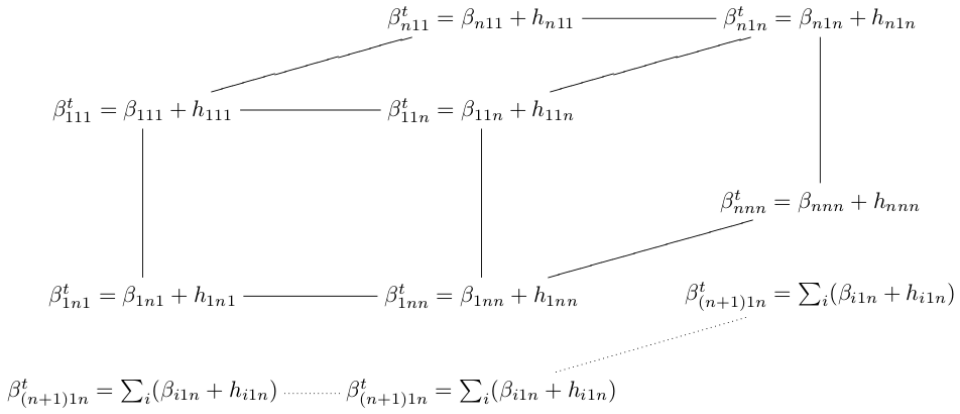

Figure 4: Users' model cube

As before, at each time instant, we update the corresponding kij-th element and the corresponding last row of the cube. The β_{kij}'s elements are assessed as above.

4.2.3 Model averaging

We describe now how model averaging and updating takes place within our model. First, recall that we shall use

$$p(b_t \mid a_t, b_{t-1}, b_{t-2}, D_t, B_u) =$$

$$= p(M_1 \mid D_t, B_u)p(b_t \mid a_t, D_t, B_u) + p(M_2 \mid D_t, B_u)p(b_t \mid b_{t-1}, b_{t-2}, D_t, B_u),$$

with

$$p(M_i \mid D_t, B_u) = \frac{p(D_t \mid M_i, B_u)p(M_i \mid B_u)}{\sum_{i=1}^{2} p(D_t \mid M_i, B_u)p(M_i \mid B_u)},\ i = 1, 2.$$

Under the assumption $p(M_1 \mid B_u) = p(M_2 \mid B_u) = \frac{1}{2}$,

$$p(M_i \mid D_t, B_u) = \frac{p(D_t \mid M_i, B_u)}{\sum_{i=1}^{2} p(D_t \mid M_i, B_u)},$$

with

$$p(D_t \mid M_i, B_u) = \int p(D_t \mid \theta_i, M_i, B_u)p(\theta_i \mid M_i, B_u)d\theta_i$$

We provide now the computations for our models:

- M_1. We have

$$p(D_t \mid M_1, B_u) = \int \ldots \int \left(\prod_{i,j} {p_{ij}}^{h_{ij}^u} \right) k \left(\prod_{i,j} {p_{ij}}^{\beta_{ij}^u - 1} \right) dp_{ij},$$

  where k is the corresponding normalization constant. Simple computations lead to

$$p(D_t \mid M_1, B_u) = \left[\frac{\prod_{i=1}^{n} \Gamma(\beta_{i1}^u)}{\Gamma(\sum_{i=1}^{n} \beta_{i1}^u)} \ldots \frac{\prod_{i=1}^{n} \Gamma(\beta_{im}^u)}{\Gamma(\sum_{i=1}^{n} \beta_{im}^u)} \right] \times$$
$$\times \left[\frac{\Gamma(\sum_{i=1}^{n} (\beta_{i1}^u + h_{i1}^u))}{\prod_{i=1}^{n} \Gamma(\beta_{i1}^u + h_{i1}^u)} \ldots \frac{\Gamma(\sum_{i=1}^{n} (\beta_{im}^u + h_{im}^u))}{\prod_{i=1}^{n} \Gamma(\beta_{im}^u + h_{im}^u)} \right].$$

  Now, if we write

$$p(D_t \mid M_1, B_u) = p_t^{1u},$$

we can see that, if at iteration $t+1$ the bot performed a_j and user B_u performed b_i, the new model probability is updated to

$$p_{t+1}^{1u} = p_t^{1u} \times \frac{\beta_{(n+1)j}^{ut}}{\beta_{ij}^{ut}}$$

- M_2. We have

$$p(D_t \mid M_2, B_u) = \int \dots \int \left(\prod_{i,j,k} {p_{ijk}}^{h_{ijk}^u} \right) k' \left(\prod_{i,j,k} {p_{ijk}}^{\beta_{ijk}^u - 1} \right) dp_{ijk}$$

where k' is the appropriate normalisation constant. Simple computations lead to

$$p(D_t \mid M_2, B_u) = \left[\prod_{i=1}^{n} \frac{\Gamma(\beta_{i11}^u)}{\Gamma(\beta_{i11}^u + h_{i11}^u)} \ \dots \ \prod_{i=1}^{n} \frac{\Gamma(\beta_{inn}^u)}{\Gamma(\beta_{inn}^u + h_{inn}^u)} \right] \times \\ \times \left[\frac{\Gamma(\sum_{i=1}^{n} (\beta_{i11}^u + h_{i11}^u))}{\Gamma(\sum_{i=1}^{n} \beta_{i11}^u)} \ \dots \ \frac{\Gamma(\sum_{i=1}^{n} (\beta_{inn}^u + h_{inn}^u))}{\Gamma(\sum_{i=1}^{n} \beta_{inn}^u)} \right].$$

Again, we may write the result recursively as follows. If we designate

$$p(D_t \mid M_2, B_u) = p_t^{2u},$$

then,

$$p_{t+1}^{2u} = p_t^{2u} \times \frac{\beta_{(n+1)jk}^{ut}}{\beta_{ijk}^{ut}},$$

assuming that, at iteration $(t+1)$, the user B_u performed b_k, after having performed b_i and b_j.

4.2.4 User identification $p(B_u)$

This is based on standard face recognition models using OpenCV libraries. We use the eigenface algorithm of recognition, see [9]. It consists on two phases: learning and recognition. In the first phase, it receives one or more face images (training images) for each user that is going to be recognized. In the second phase, once a face image is provided, it looks for the "closest" training face image and, if the distance is above a threshold, it assumes that person is recognized. Otherwise, the face is classified as an "unknown" person. We consider, thus, that the user is that which maximizes $p(B_u|D_t)$ after obtaining an image of the face of the participant, assuming that such quantity is big enough.

4.2.5 The environment model

We describe now the environment model. For illustrative purposes, we shall just consider four environmental variables, $e_t = (e_t^1, e_t^2, e_t^3, e_t^4)$, so that:

- e_t^1, refers to energy level at time t.
- e_t^2, refers to temperature at time t.
- e_t^3, refers to inclination at time t.
- e_t^4, refers to noise at time t.

We assume conditional independence for the four environmental variables, so that

$$p(e_t \mid b_t, e_{t-1}, e_{t-2}) = \prod_{i=1}^{4} p(e_t^i \mid b_t, e_{t-1}^i, e_{t-2}^i).$$

We describe now the evolution models for each of the environmental variable.

Energy level model We shall assume that $p(e_t^1 \mid b_t, e_{t-1}^1, e_{t-2}^1) = p(e_t^1 \mid b_t, e_{t-1}^1)$. We just need to know the current energy level and the action of the users (whether they just plugged in or not the bot) to forecast the energy level, or whether the bot is on charge or not. Indeed, we have that

- If $b_t \neq b_1 = recharge$ and the wire is unplugged, $e_t^1 = e_{t-1}^1 - k_1\Delta t$, where k_1 is the energy consumption rate.
- If $b_t = b_1 = recharge$ or the wire is plugged in, $e_t^1 = e_{t-1}^1 + k_2\Delta t$, where k_2 is the energy recharging rate.

Temperature model We shall assume that $p(e_t^2 \mid b_t, e_{t-1}^2, e_{t-2}^2) = p(e_t^2 \mid e_{t-1}^2, e_{t-2}^2)$, as we are not able to detect the actions of the user concerning temperature changes. We shall assume a simple model such as $e_t^2 = [e_{t-1}^2 + (e_{t-1}^2 - e_{t-2}^2)]\Delta t$.

Inclination model We shall assume the generic model $p(e_t^3 \mid b_t, e_{t-1}^3)$, being $b_t = attack$, the relevant user action. The inclination sensor detects only whether (1) or not (0) the bot is in vertical position. Then, we use the evolution matrix shown in Table 1.

e_{t-1}^3	Attack	Not attack
0	0	0
1	0	1

Table 1: Evolution of being in vertical position

Noise model We shall assume that $p(e_t^4 \mid b_t, e_{t-1}^4, e_{t-2}^4) = p(e_t^4 \mid e_{t-1}^4, e_{t-2}^4)$, as we are not able to detect the actions of the user concerning noise changes. We shall assume that $e_t^4 = e_{t-1}^4 + (e_{t-1}^4 - e_{t-2}^4)\Delta t$.

4.3 Multiobjective preference model

We describe now the preference model embedded in our agent.

4.3.1 Basic preference structure

The bot aims at satisfying five objectives, which, as in [15], are ordered hierarchically by importance, as shown in Fig. 5. They are:

- A primary objective concerning being properly charged.
- A secondary objective concerning being secure.
- A third objective concerning being taken into account by the users.
- A fourth objective concerning being accepted by the users.
- A fifth objective concerning being updated.

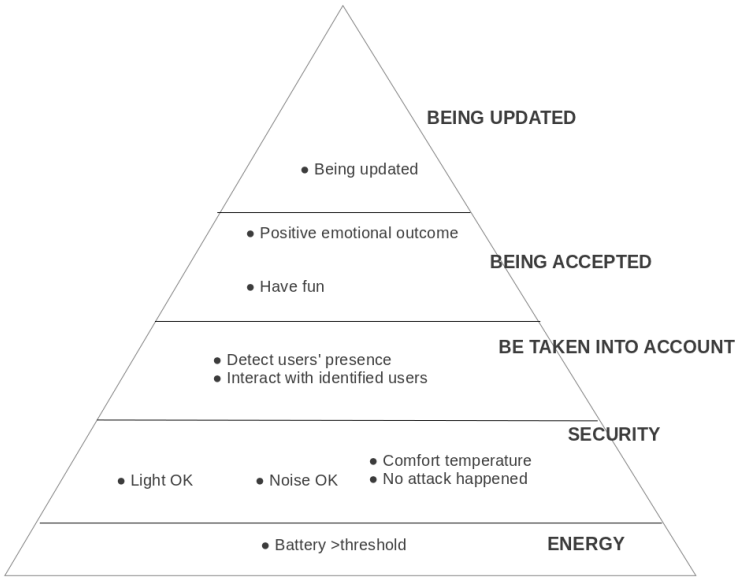

Figure 5: Objectives' pyramid

The hierarchy entails that once the bot has attained a sufficient value in a lower level objective it may devote resources to higher level objectives. This is reflected in the weights of the component utility functions. We describe now such functions.

4.3.2 Component utility functions

Energy The most basic objective pays attention only to the energy level. The bot aims at having a sufficient energy level to perform its activities. A very low energy level is perceived as bad by the bot. A sufficiently high energy level is good for the bot. We represent it through

$$u_1(ene) = \begin{cases} 0, \text{ if } ene \leq lth \\ 1, \text{ if } ene \geq uth \\ (\frac{ene-lth}{uth-lth}), \text{ otherwise,} \end{cases}$$

with $uth = 0.5$ and $lth = 0.1$

Security The second objective refers to security. It essentially takes into account whether the bot is being attacked by any user and whether it is functioning at a proper temperature. Secondarily, it pays attention to having appropriate light and noise levels in the room. It is represented through

$$u_2(attack, temp, light, noise) =$$

$$= w_{21} \times u_{21}(attack) + w_{22} \times u_{22}(temp) + w_{23} \times u_{23}(light) + w_{24} \times u_{24}(noise),$$

with $w_{2i} \geq 0$, $\sum_{i=1}^{4} w_{2i} = 1$, and weights ordered in importance as follows: $w_{21} > w_{22} > w_{23} > w_{24}$.

The component utility functions are

$$u_{21}(attack) = \begin{cases} 0, \text{ if } no\ attack\ happened \\ \\ 1, \text{ otherwise,} \end{cases}$$

$$u_{22}(temp) = \begin{cases} 0, \text{ if } \ temp < lth \ \text{ or } \ temp > uth \\ \\ 1, \text{ if } \ \ ltcth < temp \text{ or } < utcth \\ \\ 1 - (\frac{ltcth - temp}{ltcth}), \ \text{ if } \ temp < ltcth \\ \\ \frac{uth - temp}{uth - utcth}, \ \text{ if } temp > utcth, \end{cases}$$

with $lth = 0°$ C, $uth = 35°$ C, $ltcth$ (lower thermal comfort) $= 20°$ C and $utcth$ (upper thermal comfort) $= 25°$ C.

$$u_{23}(light) = \begin{cases} 0, \text{ if } \ light > uth \\ 1, \text{ if } \ \ llcth < light \ < ulcth \\ \\ 1 - (\frac{llcth - light}{llcth}), \ \text{ if } \ light < llcth \\ \\ (\frac{uth - light}{uth - ulcth}), \ \text{ if } light > ulcth, \end{cases}$$

with uth = 2000 lux, $llcth$ (lower lighting comfort) = 200 lux and $ulcth$ (upper lighting comfort) = 1000 lux.

$$u_{24}(noise) = \begin{cases} 0, \text{ if } noise > uth \\ 1, \text{ if } noise < lth \\ 1 - (\frac{noise - lth}{uth - lth}), \text{ otherwise} \end{cases}$$

with lth = 70 dB and uth = 130 dB.

Be taken into account The third objective is related with being taken into account by the users. It evaluates whether some of the owners are around it and whether they are interacting with it by asking the bot to play, ordering something, starting a conversation or, simply, not ignoring it. We represent it through the component utility function

$$u_3(interaction, detection) = w_{31} \times u_{31}(interaction) + w_{32} \times u_{32}(detection),$$

with $w_{3i} \geq 0$, $\sum_{i=1}^{2} w_{3i} = 1$, and weights ordered in importance as follows: $w_{31} >> w_{32}$.

We further decompose u_{31} as follows:

$$u_{31}(interaction) = w_{311} \times u_{311}(not\ ignored) + w_{312} \times u_{312}(be\ spoken) +$$

$$+ w_{313} \times u_{313}(asked\ to\ play) + w_{314} \times u_{314}(be\ ordered),$$

with $w_{31i} \geq 0$, $\sum_{i=1}^{4} w_{31i} = 1$, and weights ordered in importance as follows: $w_{311} > w_{312} > w_{313} > w_{314}$. The corresponding component utility functions are:

$$u_{311}(not\ ignored) = \begin{cases} 1, \text{ if } b_t \neq ignored \\ \\ 0, \text{ otherwise,} \end{cases}$$

$$u_{312}(be\ spoken) = \begin{cases} 1, \text{ if } a\ grammar\ has\ been\ initiated \\ \\ 0, \text{ otherwise,} \end{cases}$$

$$u_{313}(asked\ to\ play) = \begin{cases} 1, \text{ if } it\ is\ asked\ to\ play\ by\ the\ users \\ \\ 0, \text{ otherwise,} \end{cases}$$

$$u_{314}(\textit{being ordered}) = \begin{cases} 1, \text{ if } \textit{it receives an order} \\ \\ 0, \text{ otherwise,} \end{cases}$$

Being ordered, asked to play or *be spoken* are evaluated through an ASR algorithm. With respect to u_{32}, we shall use

$$u_{32}(\textit{detection}) = \frac{\textit{voice} + \textit{vision}}{\textit{total}}$$

being *voice* the % of voice recognition obtained by the ASR algorithm, *vision* the % of face recognition resulting from the OpenCv algorithm [9] and *total* the maximum possible value of the sum of *voice* and *vision* (200).

Being accepted The fourth objective is aimed at evaluating whether the bot is being accepted by the users, checking its emotional state and whether it is having fun with some of the users. It is represented through

$$u_4(fun, \textit{emotional outcome}) = w_{41} \times u_{41}(fun) + w_{42} \times u_{42}(\textit{emotional outcome}),$$

with $w_{4i} \geq 0$, $\sum_{i=1}^{2} w_{4i} = 1$, and weights ordered as: $w_{41} > w_{42}$.

Regarding $u_{41}(fun)$ we decompose it into

$$u_{41}(fun) = w_{411} \times u_{411}(play) + w_{412} \times u_{412}(flatter) + w_{413} \times u_{413}(stroke),$$

with $w_{41i} \geq 0$, $\sum_{i=1}^{3} w_{41i} = 1$, and weights ordered in importance as follows: $w_{411} > w_{412} > w_{413}$. The component utility functions are:

$$u_{411}(play) = \begin{cases} 1, \text{ if } b_t = play \\ \\ 0, \text{ otherwise,} \end{cases}$$

$$u_{412}(flatter) = \begin{cases} 1, \text{ if } b_t = flatter \\ \\ 0, \text{ otherwise,} \end{cases}$$

$$u_{413}(stroke) = \begin{cases} 1, \text{ if } b_t = stroke \\ \\ 0, \text{ otherwise,} \end{cases}$$

for t, $t-1$ and $t-2$.

$$u_{42}(emotional\ outcome) = \begin{cases} 1, \text{ if } \sum_t positive\ emotions > \sum_t negative\ emotions \\ \\ \frac{\sum_t negative\ emotions - \sum_t positive\ emotions}{\sum_t emotions}, \quad \text{otherwise} \end{cases}$$

for t, $t-1$ and $t-2$.

Being updated Finally, in the fifth level objective the bot is looking forward to being updated. Our current implementation of such component utility function is

$$u_5(updated) = \begin{cases} 1, \text{ if } bot's\ version\ date's\ <\ 2\ months\ ago \\ \\ 0, \text{ otherwise,} \end{cases}$$

Global utility function The bot will evaluate the impact of that action on its objectives, based on the objective tree in Fig. 6, which includes the corresponding attributes and, consequently, the relevant sensors.

Based on these five level objectives, the global utility function would be

$$w_1 \times u_1(ene) + w_2 \times u_2(attack, temp, light, noise) + w_3 \times u_3(detection, interaction) +$$

$$+ w_4 \times u_4(emotional\ outcome, fun) + w_5 \times u_5(cooperation, updated),$$

with $w_1 >> w_2 >> w_3 >> w_4 >> w_5$ and $w_1 + w_2 + w_3 + w_4 + w_5 = 1$, to stress the hierarchical nature of the objectives.

4.4 Emotional models

Emotional models are based on [8], who compare different appraisal models to obtain the intensity of an emotion, favouring weighted variations of expected utility models.

Despite the abundance of basic emotions in the literature, we have decided to simulate two positive emotions (joy and hope) and two sad emotions (sad and fear). We recalibrated the models in [8] to get a better fit with our bot.

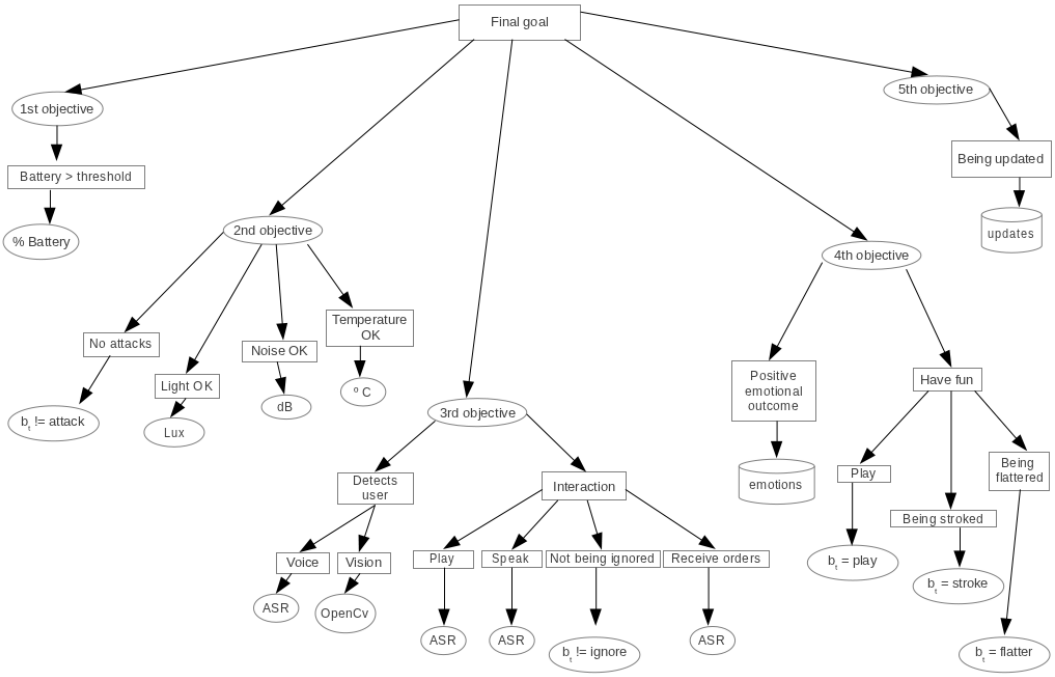

Figure 6: Objectives hierarchy

The evolution of emotions is modulated based on the decision field theory difference equations in [28]. Mixture of emotions is built as in [6] and the impact of emotions on weights is dealt with as in [29]. Emotions depend on the perceived user B_u.

4.5 Optimising expected utility

The model is implemented in an asynchronous mode. Sensors are read at fixed times (with different timings for different sensors). When relevant events are detected, the basic information processing and decision making loop, described in Fig. 1, is shot. It is managed by exception in that if exceptions to standard behavior occur, the loop is open to interventions through various threads. Given the processor in our bot, and the need to have almost instant responses, we plan only one step ahead and choose the action with probabilities proportional to the computed expected utilities, to cater for some variety. Memory is limited to the two previous instants. See the

videos [24] and [25] for some demonstration of the behavior.

5 Discussion

We have described a behavioral model of an affective agent facing several intelligent adversaries using multi-attribute decision analysis at its core, complemented by forecasting models of the adversaries (Adversarial Risk Analysis) and emotion-based behavior (Affective Decision Making). We have implemented the model in a eduitainment bot. Improving user's experience when interacting with a bot, [3] or [13], was our motivation for this model, but we have realized that it may find many other potential applications in fields like interface design, e-learning, entertainment or as therapeutical devices through artificial pets for the elderly or kids with cognitive problems.

As future work, we are thinking about extending the model to a case in which the agent cooperates or competes, depending on its emotional state, with other agents looking to accomplish a social goal. Dealing with the possibility of learning about new users' actions, based on repeated readings, and, consequently, augmenting the set $\mathcal{B}$ is another challenging problem. Also, we contemplate incorporating addtional emotions. Finally, we have used what is termed a 0-level ARA analysis. We could try to undertake higher ARA levels in modeling the performance of adversaries.

Acknowledgments

Research supported by grants from the MICINN project RIESGOS, the RIESGOS-CM project and the INNPACTO project HAUS. We are grateful to discussion with Diego Garcia, from AISoy Robotics, Jesus Rios, from IBM Research and David Banks, from Duke University.

References

1 Banks, D., Petralia, F., Wang, S. (2011), Adversarial risk analysis: Borel games. *Applied Stochastic Models in Business and Industry*, **27**: 72-86.

2 Bellman, R. (1957) *Dynamic Programming*. Princeton University Press, Princeton, NJ.

3 Breazeal, C. (2002) *Designing Sociable Robots.* The MIT Press.

4 Clemen, R.T., Reilly, T. (2004) *Making Hard Decisions with Decision Tools.* Duxbury: Pacific Grove, CA.

5 Damasio, A. R. (1994) *Descartes' Error: Emotion, Reason, and the Human Brain.* New York: G.P. Putnam.

6 El-Nasr, M.S., Yen, J., Ioerger, T.R. (2000) FLAME: Fuzzy Logic Adaptive Model of Emotions. *Autonomous Agents and Multi-Agent Systems* **3**(3):219-257.

7 Glimcher, P. W., Camerer, C., Poldrack, R. A., Fehr, E. (2008). *Neuroeconomics: Decision Making and the Brain*, Academic Press.

8 Gratch, J., Marsella, S., Wang, N., Stankovic, B. (2009) Assessing the validity of appraisal-based models of emotion. In Pantic, M., Nijholt, A., Cohn, J. (eds.), *Proceedings of the International Conference on Affective Computing and Intelligent Interaction.* Amsterdam, Netherlands, ACII'09: IEEE Computer Society Press.

9 Hewitt, R. Seeing With OpenCV, Part 4: Face Recognition With Eigenface. *SERVO Magazine, April 2007.* Retrieved September 16, 2010, from: $www.cognotics.com/opencv/servo_2007_series/part_4/index.html$.

10 Hoeting, J., Madigan, D., Raftery, A., Volinsky, C. (1999) Bayesian model averaging: A tutorial, *Statistical Science*, **4**, 382-417.

11 Kadane, J. B. (2011). Adversarial Risk Analysis: What's new, what isn't?: Discussion of Adversarial Risk Analysis: Borel Games. *Journal Applied Stochastic Models in Business and Industry*, **27**, 2 (March 2011), 87-88.

12 Kadane, J. B., Larkey, P. D. (1982) Subjective probability and the theory of games. *Management Sci* **28**(2):113-120.

13 Kirby, R., Forlizzi, J., Simmons, R. (2010) Affective social robots. *Robotics and Autonomous Systems* **58** 3:322-332.

14 Loewenstein, G.(1996) Out of Control: Visceral Influences on Behavior. *Organizational Behavior and Human Decision Processes*, **65**(3):272-292. Carnegie Mellon University.

15 Maslow, A. H. (1943) A theory of human motivation. *Psychological Review*, **50**, 4, 370-96.

16 Paté-Cornell, M. E. and Guikema, S. O. (2002). Probabilistic Modeling of Terrorist Threats: A Systems Analysis Approach to Setting Priorities Among Counter-measures. *Military Operation Research*, 7, 5-23.

17 Picard, R. W. (1997) *Affective Computing.* Cambridge, MA: MIT Press

18 Raiffa, H. (2007) *Negotiation Analysis: The Science and Art of Collaborative Decision Making.* Cambridge, Massachusetts:Belknap Press of Harvard University Press

19 Ríos Insua, D., Ríos, J., Banks, D. (2009) Adversarial risk analysis. *Journal of the American Statistical Association* **104**(486):841-854.

20 Ríos Insua, D., Ruggeri, F., Wiper, M. (2012) *Bayesian Analysis of Stochastic Process Models*, Wiley.

21 Ríos Insua, D., Salewicz, K. (1995) The operation of Kariba Lake: a multiobjective decision analysis. *Journal of Multicriteria Decision Analysis*, 1995, 4, 203-222.

22 Stahl, D. O. and Wilson, P. W. (1995). On Players Models of Other Players: Theory and Experimental Evidence. *Games and Economic Behavior*, 10(1): 218254.

23 Velásquez, J. D. (1997) Modeling Emotion and Other Motivations in Synthetic Agents. *Proceedings, 14th National Conference on AI*, AAAI Press.

24 Youtube video: *http://youtu.be/No7MqxUONRs.*

25 Youtube video: *http://youtu.be/HVJUtLxWKw4.*

26 West, M., Harrison, P. J. (1997) *Bayesian Forecasting and Dynamic Models.* New York: Springer.

27 von Winterfeldt, D., Edwards, W. (1986). *Decision Analysis and Behavioral Research.* New York: Cambridge University Press.

28 Busemeyer, J. R., Dimperio, E., Jessup, R. K. (2006) *Integrating emotional processes into decision-making models* in Gray (Ed.) Integrated models of cognitive systems, Oxford University Press.

29 Sribhashyam, S., Montibeller, G. (2012) Modeling State - Dependent Priorities of Malicious Agents. To appear in *Decision Analysis*.