Affective Knowledge Representation (AKR) for Cooperative Affective Robots (CARS)

Christine L. Lisetti Demetrios G. Glinos

Department of Computer Science
University of Central Florida
Orlando, Florida 32816-2362
lisetti@cs.ucf.edu glinosd@saic.com

Robin R. Murphy Russ Tardif

Department of Computer Science and Engineering
University of South Florida
Tampa, Florida 33620-5399
murphy@csee.usf.edu tardif@csee.usf.edu

Abstract

In this article, we describe an Affective Knowledge Representation (AKR) scheme to represent emotion schemata to be used in the design a variety of socially intelligent artificial agents. Our approach in this article and in the applications of our AKR scheme, focuses on the notion of ``social expertise'' of socially intelligent agents in terms of their 1) external behavior and 2) internal motivational goal-based abilities. We claim that social expertise will be critical for the success of human-robot interaction in a variety of applications [6] and that affective phenomena such as emotions and personality are essential in terms of social expertise and autonomous behavior. AKR includes a taxonomy of affect, mood, emotion, and personality, as well as a framework for emotional state dynamics for autonomous behavior. AKR model is being applied to design and implement two collaborating robots which exhibit social expertise as specified above. We have also developed a simulator for our collaborative affective robots in order to generate performance metrics for both real and simulated environments, the Cooperative Affective Robot Simulator (CARS) which we currently describe. CARS is a reconfigurable Java-based simulator for rapid prototyping of control algorithms for cooperating affective robotic agents and it is a tool for evaluating the performance of our autonomous cooperative social robots. In particular, the simulator was configured to represent the robotic agents at the AAAI Mobile Robot Competition's *Hors D'Oeuvres, Anyone?* event.

Introduction

Robotic agents have been the subject of intense interest in the field of Artificial Intelligence for over 40 years [1]. More recently, systems of cooperating social robotic agents have been the subject of research interest. Murphy (2002) reviews prior work on multi-robot teams and reports in detail on the authors' combined USF-UCF winning entry of the *Nils Nilsson Award* at the 2000 AAAI Mobile Robot Competition's *Hors D'Oeuvres, Anyone?* event, involving a team of two robots – a waiter and a refiller – cooperating to serve hors d'oeuvres at a reception and to converse with the crowd.

Although a relatively new field, affective computing – defined as computing that "relates to, arises from, or deliberately influences emotions" (Picard, 1997) – has established a firm foothold within the AI community. Indeed, the field has progressed to the point where Lisetti (2002) has proposed a comprehensive taxonomy reconciling and combining the principal constructs of the main current theories of affect, mood, and emotion, the Affective Knowledge Representation (AKR) scheme, which we partially applied for the design of our social robots. Moreover, Lisetti (2002) has also reviewed the case for including affect as an essential component in rational decision making through its influence in focusing attention, eliminating deadlock, and otherwise permitting decisions to be made in less than optimal conditions, and proposed a formalism for including emotions explicitly in the design of artificial rational agents.

The Cooperative Affective Robot Simulator (CARS) was conceived (1) as a reconfigurable means for the rapid prototyping of control algorithms for our social robotic agents endowed with increasingly more complex affective knowledge (AKR), and (2) as a tool for generating performance metrics and evaluate our algorithms for socially intelligent autonomous robots.

Related Research

A review of the pertinent research literature has not disclosed any reports of simulators for cooperating affective agents. As such, we believe that our efforts here constitute the first report combining constructive simulation and cooperative affective robotic agent design and performance analysis. Nevertheless, cooperating mobile robots have been studied extensively. Murphy (2002)] reviews some of the research efforts most pertinent to affective agents, and presents the authors' own robotic agent designs, which explicitly included affect.

Murphy's (2002) agent architecture incorporates affect through an Emotion State Generator (ESG) and a Behavior State Generator (BSG). The ESG determines the agent's

affective state based on various task progress measures, which also influences the agent's behavior as determined by the BSG, which operates on both such progress measures and the current affective state. This architecture was implemented in a cooperating team of two robots, "Butler and "Leguin" , who were programmed to serve the different roles of waiter and refiller, respectively, for serving hors d'oeuvres at the 2000 AAAI Mobile Robot Competition's *Hors D'Oeuvres, Anyone?* event, at which they won the coveted Nils Nilsson Award for Integrating AI Technologies. The same robot team was also demonstrated at Tampa's Museum of Science and Industry (MOSI), performing the similar task of distributing pencils and informational literature.

We followed Murphy and Lisetti's [11] separation of the implementations of affective state and behavior determination in separate and distinct modules. By separating these key aspects of agent performance, the influence of affect on behavior, and the more indirect influence of behavior on affect (through generation of different task progress measures), can be studied rigorously. We have therefore adopted this separation as a requirement for the external interface of a CARS robot, although of course, such robot may combine these components internally if so desired.

We have also selected Murphy's Butler and Leguin as the first application for the simulator. Using the robot specifications described by Murphy and Lisetti [11], we developed their CARS analogues, with only minor variations. These experiments are described in a later section of this paper.

Affect and Emotions for Social Artificial Agents

Affective Knowledge Representation (AKR)

The functional role of emotions has been recently fully recognized as essential for intelligent agents with limited resources operating in an complex and unpredictable environment (Murphy et. Al, 2002). In order to contribute to rendering artificial intelligent agents socially more competent, we combined and reconciled aspects of the main current theories of affect, mood and emotion (Ortony et al. 1988), (Frijda, 1986), (Wierzbicka, 1992), into a simplified comprehensive, (but not complete) taxonomy of affect, mood and emotion for computational Affective Knowledge Representation (AKR).

We created AKR to enable the design of a variety of artificial autonomous (i.e. self-motivated) socially competent agents, from robotics (Murphy et al. 2002), (Lisetti, 1997), to user-modeling (Lisetti and Bianchi 2002), to human-computer interaction (Hayes-Roth et al. 1998), to multi-agent systems and distributed AI. Our taxonomy of affective states in Figure 1 is aimed at differentiating among the variety of affective states by using values of well-defined componential attributes. First we define our use of terms throughout our work.

In short, in our taxonomy, each emotion is considered as a collection of emotion components, such as its valence (the pleasant or unpleasant dimension), its intensity (mild, high, extreme), etc. In our representation, we also included the action tendency of each emotion (Frijda, 1986) which corresponds to the signal that the emotional state experienced points to: a small and distinctive suite of action plans that has been (evolutionarily) selected as appropriate, e.g. approach, avoid, reject, continue, change strategy, etc.

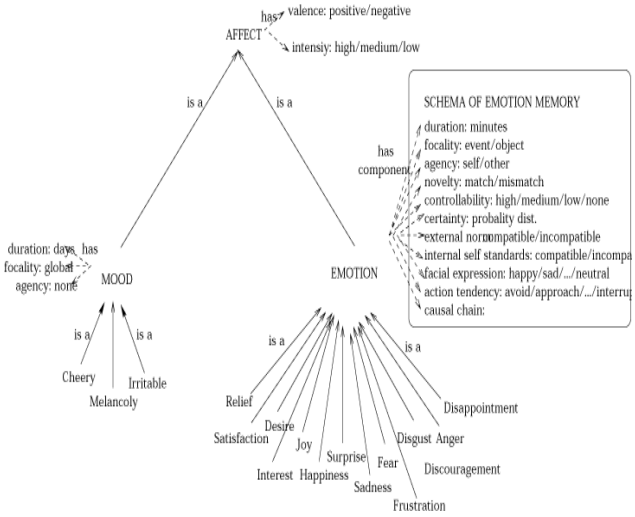

Figure 1: Affective Knowledge Representation (AKR)

Personality (not drawn): We identify personality as representing characteristics of an autonomous organism that account for consistently chosen patterns of mental reaction including behavior, emotions and thoughts over situations and time (Moffat, 1997). Furthermore, unlike affective and emotive states which are short-lived, personality lasts a life-time (albeit personality disorders occur). In our example, only one trait – *caution* – is represented, and it affects how "mercurian" the robot's intensity of emotions will vary in terms of the robot's progress toward its goal.

Affect: Affect varies along two dimensions: (i) *valence* which can be positive or negative (the pleasant and unpleasant dimension) and (ii) *intensity* which varies in terms of degree. Affect considered as a state is therefore a coarse-grained phenomena, and because of the hierarchical structure of affective phenomena as a whole, it can also be

semantically considered as including finer-grained states as moods and emotions as shown in Figure 1.

Mood: Moods are affective phenomena encoding coarser-grained information and of shorter duration than emotions. In addition, moods are not typically elicited by any specific stimulus, and therefore have a global focality, and no specific agency associated with them.

Emotion: We identify emotions as changes in activation of behavioral dispositions or transformation of dispositions to act, caused by relevant events or circumstances.

Because emotions are at the bottom of the hierarchical model, emotions do not necessarily imply personalities, since some emotions might be experienced by different types of agents (artificial or natural). The type of personality, is inherited from the higher node in the tree, allowing agents of different personality type to still experience the full range of possible emotions as advocated by other computational approaches (Castelfranchi, 1997).

Furthermore, because we adopt the functional view of emotions, which identifies emotions as related to goals and action tendencies (such as self-preservation, avoid negative experiences, approach enjoyable things, etc.), our model is compatible with goal-oriented theory of personality (Carbonnell 1980).

In addition, because the interactive strategies (Tit-for-Tat, cheat, etc.) and preferences (avoid/dislike negative and seek/like positive affect) are specified in the model at a higher level than at the emotion level, and because personality is explicitly represented as lasting a lifetime and not related with any specific event, this approach is in agreement with other views which emphasize the main distinction between personality (stable) and emotion (changeable). Positive and negative affect can also be considered equivalent to positive and negative attitude, while personality traits are one component of the entire personality description). This approach is also consistent with the idea that different personalities can influence an agent's propensity toward a particular set of emotions and moods.

Emotion Components

In order to address some of the difficulties of the previous computational approaches pointed out by Pfeifer (Pfeifer 1988) -- namely the lack of representation of physiological and subjective parameters -- we do not split 'emotion' and 'cognition', but rather merge them into a structure that encapsulates simultaneously each of the three phenomena accompanying emotions:

(1) autonomic nervous system (ANS, henceforth) arousal, signaling affect intensity and valence;
(2) expression, for now only facial expression (but could also include vocal, and body posture); and
(3) subjective experience, including cognitive appraisals (such as modal beliefs, criteria, standards, etc.).

In an effort to identify what makes one emotion different from another, we include elements from the "cognitive view" of emotions, which advocates a componential approach (Leventhal, 1987), (Frijda, 1986), (Ortony, 1988) (Roseman, 1996). From this approach, cognitive structures associated with emotions are considered to represent the subject's checks (appraisal or evaluation) of the events confronting them.

These checks are part of the subjective experience of emotion and can be represented with a limited number of components. Each type of checks is described as a unique pattern of such components, or dimension values. As with the set of basic emotions which varies among theories, several dimensions are often considered, but the following are found in most analyses: valence (pleasantness or unpleasantness), intensity/urgency, agency/responsibility, novelty, controllability, modifiability, certainty, external social norms compatibility, and internal standards compatibility. We also included duration and focality which differentiate emotions from moods, for future potential expansion of the model. A full description of the AKR parameters can be found in (Lisetti 2002), and we currently explain only a subset of the components which related to the design of our autonomous mobile social robots:

facial expression *(happy/sad/surprised/disgusted/angry /fearful/ neutral):* is used to store the facial expression associated with the current emotion. Some emotions are not associated with any specific facial expression (neutral), or can vary among cultures and individuals. We use stored facial expressions – culture independent emoticons (Lewitt, 1992) – to quickly reflect externally the current inner states of our artificial agents to humans, in a similar fashion as human facial expressions are used in social interactions to quickly communicate inner states to others.

valence *(positive/negative)*: is used to describe the pleasant or unpleasant dimension of an affective state. Each affective is associated with a valence, except for the emotion of surprise which can be either positive or negative depending upon the nature of the stimulus. In our case, valence is determined in terms of the robot's progress toward their respective goals to cooperate serving guests (described later).

intensity *(very high/high/medium/low/very low*): varies in terms of degree (and can also be internally represented with scalars). The intensity of an affective state is relevant to the importance, relevance and urgency of the message

that the state carries. In natural organisms, valence and intensity is signaled by the activity of the autonomic nervous system, along the physiological dimension generated by the body proper, and do not necessarily involve the cognitive apparatus. In our case, intensity is function of the measure of the robots progress, which is affected by a overall *caution* personality trait: the more cautious the agent, the faster the intensity of emotions rises.

duration *(lifetime/days/minutes)*: is used to indicate that moods are more muted and last longer than emotions, which is indicated by the duration attribute measured in terms of days, as opposed to minutes in the case of emotions; it can also be used to resolve the conflict between personality and emotion by assuming that the underlying mechanisms are essentially the same, and that only the time-course and temporal consistency of their influence varies: personalities can be permanent and last a lifetime. Since we only deal with emotions at this stage of our implementation, duration is set as a default to minutes.

agency (*self/other)*: is used to indicate who was responsible for the emotion, the agent itself *self*, or someone else *other*. In our example, if the waiter is frustrated at its assistant, the agency parameter will be 'assistant'; but if the agent is happy, agency will point to 'self'.

action tendency: identifies the most appropriate (suite of) actions to be taken from that emotional state (described next).

causal chain: identifies the causation of a stimulus event (described later).

Functional Attributes and Action Tendencies

From the Darwinian categorical theory of emotions, emotions can be discretely categorized. Emotions are considered as mental and physiological processes, caused by the perception of general categories of event, that elicits internal and external signals and matching suite of action plans or *causal chains* (described next). The Darwinian perspective proposes that bridging the gaps of rationality becomes possible if many specific emotional states are mapped into a few broad classes of reaction, or *action tendencies*.

action tendency: Emotions which are called "primary" or "basic" are such in the sense that they are considered to correspond to distinct and elementary forms of action tendency. Each ``discrete emotion'' calls into readiness a small and distinctive suite of action plans that has been selected as appropriate when in the current emotional state. Thus in broadly defined recurring circumstances that are relevant to goals, each emotion prompts both the individual and the group in a way that has been evolutionarily more successful than alternative kinds of prompting.

The number and choice of what is called basic or primary emotions vary among various emotion theories, and we have selected the ones that seem to reoccur consistently across emotion theories. Their associated action tendencies are listed in the Table 1.

ACTION TENDENCY	FUNCTION	EMOTION
Avoid	Protect	Fear
Approach	Permit consummatory Activity	Desire
Agnostic	Regain Control	Anger
Reject	Protect	Disgust
Prepare	Caution	Anxiety
Inactivity	Recuperation	Contentment

Table 1: Action Tendencies

The emotional signal that is sent when a subgoal is achieved acts to prompt the individual to continue with the current direction of action, whereas the signal sent when a goal is lost, indicates a need to change the course of action, or to disengage from the goal. Ensuing actions can be communicated to others in the same social group, which in turn, can have emotional consequences for these other individuals too.

For our specific current application, the robot waiter action tendencies (AT) associated with of its emotion are related to their respective tasks and shown in Table 2.

ACTION TENDENCY	AT for Waiter /Refiller	EMOTION
FreeActivate	Serve/Joke	Happy
ContinueNormal Activity	Serve;CallforRefill /Serve	Confident
MonitorProgress Closely	AsktoHurry /IncreaseSpeed	Concerned
ChangeCurrent Strategy	Intercept /GoHome	Frustrated

Table 2: Waiter Robot Action Tendencies

Emotion Beliefs and Causal Chains

We adapted the semantic meta-definitions of emotion concepts developed by Wierzbicka using language independent primitives (Wierzbicka, 1992) to create the *causal chain*.

Causal chain: A causal chain of events describes the subjective cognitive experience components which are associated with the emotion, the beliefs, the goals, and their acheivement or lack of. These are associated with each emotion and can be spoken via speech synthesis for the agent to *verbally* express and describe the cognitive interpretation of its state. Examples can be found in (Lisetti, 2002).

Emotional States Dynamics

Our AKR framework also establishes a dynamic model of emotional states for our robots architecture which is based on the multi-level process theory of emotions (Leventhal and Scherer 1987) shown in Figure 2.

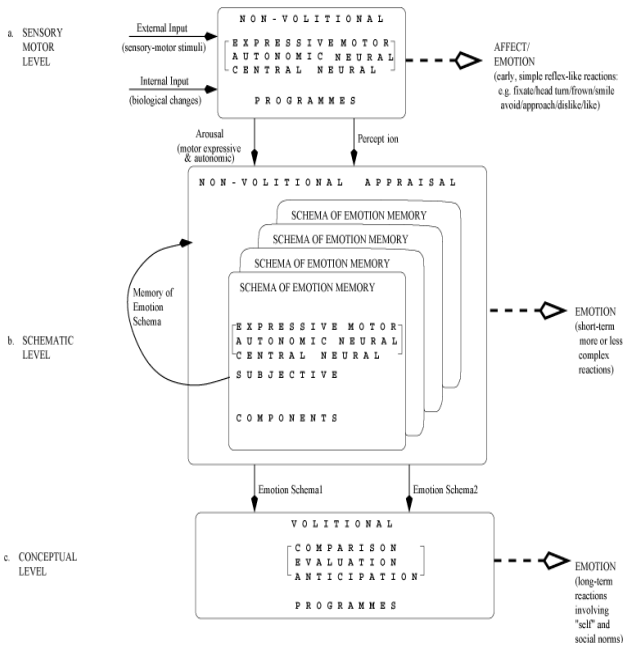

Figure 2: Multi-level Process Affect/Emotion Generation

In short, the multi-level process theory of emotions considers emotion as complex behavioral reactions constructed from the activity of a hierarchical multi-component processing system:

(1) the *sensory-motor level* is activated automatically without deliberate planning by a variety of external stimuli and by internal changes (e.g. hormonal levels). Affective reactions based on pure sensory-motor processes are reflex-like and are coarse-grained states as described above: information available at that level consists of valence and intensity;

(2) the *schematic level* integrates sensory-motor processes with prototypes of emotional situations having concrete schematic representations.

(3) The conceptual level is deliberative and involves reasoning over the past, projecting into the future, comparing emotional schemata to avoid unsuccessful emotional situations.

The multi-level process theory of emotions is particularly powerful for artificial intelligent design is that it enables various levels to be implemented, integrated, and tested incrementally. It furthermore matches closely hybrid/reactive architectures for robotic agents. Table 3 shows that relationship:

Sensory-motor • emotions modify the motor outputs of active behavior	**Reactive behavioral** • active behaviors couple sensors and motor actions
Schematic • emotions control which behaviors are active through prototypical schemata • can be implemented with scripts (Lisetti, 1997)	**Assemblages of behaviors** • collections of behaviors are assembled into a prototypical schema or skill (Arkin 1990) • can be implemented with scripts (Murphy, 1996)
Conceptual • reasons about past, present emotions and projects into the future possible consequences of action from anticipated emotion (Lisetti et al., 2002)	**Deliberative Planning** • reasons about past, present, future

Table 3: Multi-level process of emotions vs. Hybrid reactive/deliberative architectures

External Event as Inputs: In our example, one of our robots has four emotional states: HAPPY, CONCERNED, FRUSTRATED and ANGRY (with HAPPY as the starting point). The transitions among the states are caused by environmental inputs or responses of the system, and they are divided into categories of positive progress toward goal and negative progress. Using this dynamic model we can predict that an agent that is in a HAPPY state will remain HAPPY given positive inputs, will become CONCERNED given negative inputs toward its goal (e.g. obstacles of some sort dependent on the context), increasingly so until it reaches the FRUSTRATED state.

At which point it can receive positive inputs and move back to a CONCERNED or HAPPY state (depending on the importance of the positive input).

The process of emotion-based control is shown in Figure 3 and described in full details in Murphy (2002).

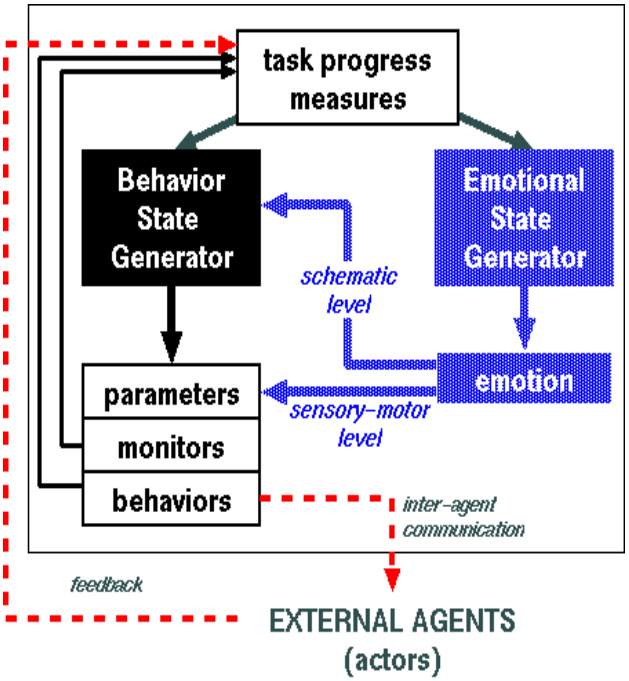

Figure 3: Emotion-based Control for Robotic Agents

Internal Beliefs as Inputs: An individual's emotions can change in regard to an event, and these changes may be the result of their own efforts, not simply the result of an independent process directed by external events or social rules. Emotional changes indeed occur as a result of a number of processes, involving emotion dynamics rather than simply outside circumstances or the force of culture.

A simple example where a negative internal belief such as ``I can't do this'' would keep the agent in its current DISCOURAGED state forever. Should the agent manage to change its internal belief for a positive input (e.g. ``I can indeed do this''), it would switch to a state of HOPEFULNESS (not shown). Other examples of such internal self-adjustments abound [9].

CARS Simulator Design

We now describe the simulator we have developed to enable further performance metrics and evaluation of our AKR for autonomous socially intelligent agents. Our approach was to duplicate the emotion-based algorithm described in [11] and shown in Figure 3, and develop a general purpose simulator for it in order to permit further testing and integration of the multi-level process of emotion-based architecture.

CARS supports a variable number of simulated robotic agents and rapid reconfiguration between simulation runs through (1) a common robot interface that permits specification of each robot's algorithms without regard either to the details of the simulator or of other cooperating robots and which supports simple inter-robot communication; and (2) a robust simulator architecture, which is easily reconfigured between successive simulation runs through a simple configuration text file, and which automatically loads the Java class files for the participating robots and configures its graphical user interface accordingly.

Given, then, that systems of cooperating robotic agents are interesting and useful for study, the question arises how best to develop such systems. One method would be to acquire robotic hardware, program it, and conduct experiments in an iterative fashion to develop final form control algorithms. But sometimes this route may not be desirable or feasible for various reasons such as the cost or availability of the hardware, or because the decision whether to invest in hardware depends, for example, on developing prototypical control algorithms of sufficient maturity that the type of hardware to acquire may be determined.

Constructive simulations, in which simulated actors operate in a simulated environment have a long history of contributing to the design optimization and performance analysis of complex systems (see, for example, [1]). Although time and effort are involved in simulator development, the costs are often far lower than the costs of prototyping hardware, and in any event development costs can often be leveraged through the development of simulators that can be reconfigured and used for multiple purposes.

Design Approach

The principal design goal for CARS was to provide a flexible and general test bed for investigating cooperative affective behavior. To this end, CARS needs to be capable of exercising a variable number of simulated robotic agents. Moreover, different robotic interaction scenarios would no doubt involve different state variables specific to their problem domains. To implement these features, the CARS functional architecture is designed so that each robot's state variables and the details of displaying its affective and behavioral state are determined entirely by the affective and behavioral modules that represent that robot, not by the simulator infrastructure.

Overall Architecture

The top-level architecture of CARS is depicted in Figure 4, which shows multiple simulated robotic agents (RAs) interacting with the external environment through a central Simulation Control (SCM) module. The simulator also includes a File System Interface (FSI) module to handle the output of log data.

The External Environment (EEM) module represents the playing field and the fixed and moving obstacles in the robotic interaction environment. For the hors d'oeuvres serving scenario, for example, there are two simulated robotic agents, representing Butler and Leguin, respectively, and an external environment representing the fixed hors d'oeuvres table and the moving people in the reception hall. The simulation then runs against the motions of the robots and moving obstacles as they are generated dynamically.

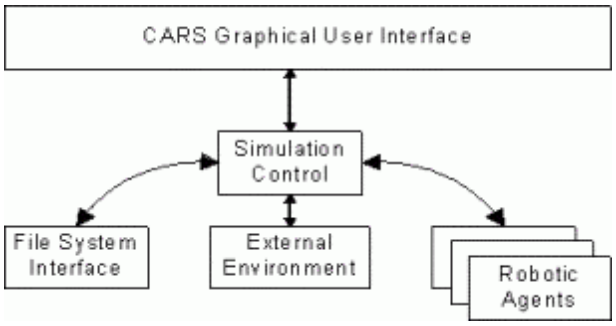

Figure 4: CARS top-level software architecture

The Graphical User Interface (GUI) module contains the simulation runtime controls and the various displays. The runtime controls permit the user to specify the size of the playing field, the number and movement specifications for moving obstacles, place fixed obstacles on the playing field, set the initial positions of the robotic agents, and set the output log file, if any. The GUI also contains controls for simulation display speed and CD-player style buttons for initializing, starting, stopping, and pausing the simulation. The GUI contains a plan view display area for displaying a "map" of the positions of all agents and obstacles, as well as separate display regions for each robotic agent to dynamically display its affective and behavioral states, and also the values of the key state variables upon which these states are computed. The GUI also contains a scrolling log display, which reports key simulation state changes as they occur. If output logging is enabled, this information is also recorded in the output log file.

Simulation Control

When the user presses the "init" control button, CARS initializes all modules. This involves resetting all robots by executing their initialization methods, initializing the set of moving obstacles according to the user-specified parameters, resetting the simulation clock to zero and sending appropriate initialization messages to the output logger. Thereafter, when the user presses the "start" button (either initially or following a pause), CARS enters the simulation control cycle shown in Figure 5. With each advance of the simulation clock, the positions of all robots and moving obstacles are updated according to their current movement specifications. Robotic agent state variables are updated based on the previous physical state and the current affective and behavioral states of the robotic agents. The updated physical state is then displayed in the plan view display component of the CARS GUI. Once updated, the affect generation components of each simulated robotic agent are exercised to determine that agent's current affective state. Finally, the behavioral change components of each simulated robotic agent is exercised to determine any behavioral state changes and any associated state variable changes.

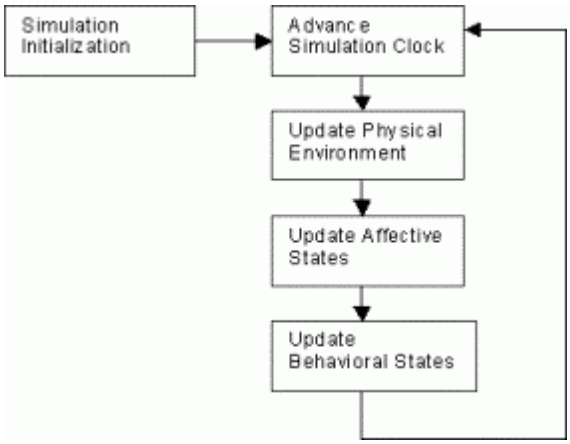

Figure 5: CARS simulation control cycle

Each simulated robotic agent possesses appropriate display methods for its affective and behavioral states and variables as it may wish to report to the interactive user. A key element of CARS's design is that the simulation infrastructure neither knows nor cares about either the variables maintained by any of the robots or what, if anything, a particular robot chooses to display. These are entirely within the control of each robot, so that a robotic agent with, for example, four states and five variables will be able to display them alongside another agent with different numbers and names of states, and with different

state variable names and values. This permits tailoring a robotic agent's display in CARS to match the corresponding fielded system's requirements, in our case for example, to match the needs of the Butler/Leguin robot development teams. Each agent displays its state information in its own panel of the GUI. The log output panel of the GUI records all state changes reported to it by each robot, for visual confirmation to the viewer, and also to a log file for later post-processing, if selected.

Visual Interface

Figure 6 shows a snapshot from this experimental configuration in progress. The control panel appears in the lower right of the figure. On the playing field in the upper right, the fixed obstacle (shown as a black square) represents the food station. The green circles represent the moving obstacles (guests). The robots are shown as red circles enclosing the first two letters of their names. The simulation log scrolls in the pane at the lower left. The upper left area contains a separate panel for each of the robots. Although the two robot status panels share stylistic features, this is not required, and even in these displays the differences in affective and behavioral states, and in reported state variables, is evident.

The snapshot shows that Butler's supply of treats is down to seven, that Butler is concerned, but that Butler is continuing to serve treats. Leguin's display shows that Leguin is also concerned, having received a "hurry" message, so that she is in her "refill" behavior. The playing field display shows that Leguin is about halfway from her starting point to Butler's location.

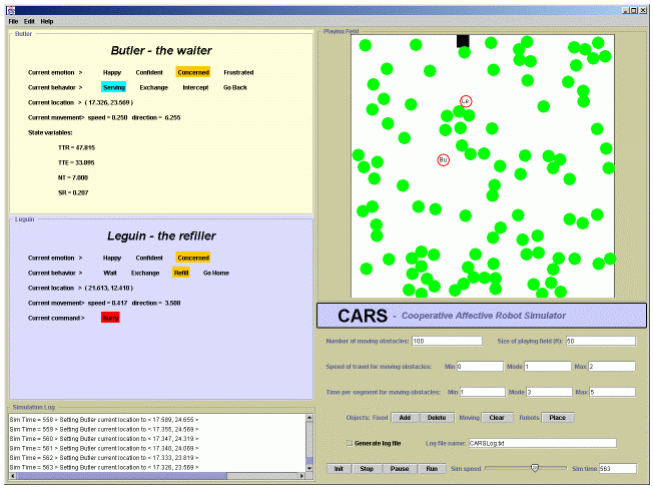

Figure 6: Snapshot of CARS simulation of waiter/refiller experiment

Inter-Robot Communication

CARS enforces a standard programming interface for simulated robots, so that both the simulation and cooperating robots may address a given robot given only that robot's name. This common robot interface (CRI) is provided by CARSRobotInterface, which is implemented as a conventional Java interface. Figure 7 contains a listing of the interface, showing all required methods and return types.

Communication between robots participating in a simulation is achieved through instance methods of the simulation Java frame object that correspond to the methods in the CRI that are not reserved for the sole use of the simulation executive. To invoke one of these methods, an additional parameter is used, corresponding to the name of the target robot whose corresponding CRI method is to be executed.
The following methods are available for inter-robot communication:

```
setRobotCommand(str,cmd)
getRobotLocation(str)
getRobotSpeed(str)
getRobotDirection(str)
getRobotStateVariables(str)
getRobotEmotionState(str)
getRobotBehaviorState(str)
```

where "str" is the name of the robot to query and "cmd" is the command to send.

Using these methods, a robot may obtain the current location, motion parameters, affective and behavior states, and a hashtable containing all internal state variables that the target robot wishes to provide to external agents. In this manner, each robot controls what others may learn about its internal workings, and the simulation infrastructure can operate without knowledge of any robot's particular state variables.

```
public interface CARSRobotInterface
{
   public void initRobot(); // Initialize robot state variables
   public String getName(); // Get the robot's name
   public double getSpeed(); // Get the magnitude of the
current velocity
   public double getDirection(); // Get the current direction of
motion
   public void updateStateVariables( int time ); // Update
state variables
   public Hashtable getStateVariables(); // Get the set of
state variables
   public Point2D.Double getCurrentLocation(); // Get the
robot's current location
   public Point2D.Double projectLocation( int time ); // Project
the robot's next location
   public void setCurrentLocation( int time, Point2D.Double p
); // Set the robot's current location
   public void setCommand( int time, String cmd); // Set the
current command for this robot
   public void updateEmotionState( int time ); // Update the
robot's current emotion state
   public String getEmotionState(); // Get the robot's current
emotion state
   public void updateBehaviorState( int time ); // Update the
robot's current behavior state
   public String getBehaviorState(); // Get the robot's current
behavior state
} // end interface CARSRobotInterface
```

Figure 7. CARS Common Robot Interface

Reconfiguration

A key design feature of the CARS software architecture is its ability to accommodate different sets and different numbers of cooperating robotic agents. This is accomplished as follows:

(1) Each robot in the set must be implemented as a single Java class declared to be in the "cars" package. The robot class must extend JPanel and implement the CARSRobotInterface programming interface. The robot class file name should be unique.

(2) Each robot must provide its own Java default constructor and paintComponent method. Robots may use any state variables they desire and may report any information desired in their paintComponent method. The paintComponent method should take into account the available space for display (obtainable through the getBounds() method), as the space available depends on the number of robots participating in the simulation.

(3) The class names for the robots participating in the simulation should be listed in the CARSConfig.txt file, one to a line. Note that where the robot name (i.e., what is returned by the robot's getName() method) differs from the class name, it is the class name that must be listed, although the simulation and the other robots will use the robot's name for communication and display purposes.

(4) The robot files should be compiled and the resultant class files should be placed in the directory from which the simulation is executed.

Obstacles

CARS simulates the interaction of robotic agents among each other and with both fixed and moving obstacles on the playing field. For ease in implementation, CARS currently models all objects, whether robotic or obstacle, as occupying a 2 foot by 2 foot space. Fixed obstacles are placed by the user using the GUI and may be placed in adjacent locations so that large and complex shaped obstacles may be constructed.

CARS creates and places moving obstacles randomly upon the playing field, and instantiates their speed and duration characteristics by triangular distributions whose minimum, mode, and maximum are specified by the user using the GUI. The initial direction of motion is also selected uniformly in the range of zero to 2Π. Once initialized, a moving obstacle is advanced along its initial velocity vector until its motion duration is exceeded, whereupon a new direction, velocity, and duration are selected and the process is repeated.

With each time step, CARS projects the current motion for all robots and moving obstacles and detects collisions with each other and with fixed obstacles. If a collision is detected, then the offending object is not advanced to the proposed location. CARS specifically leaves it up to robots themselves to determine what action to take in such event and does not presuppose any particular movement strategy.

Experimental Results

CARS was configured to represent Murphy's (2002) Butler and Leguin hors d'oeuvres serving waiter and refiller robotic team. Leguin (the refiller) is stationed initially at the food table, while Butler (the waiter) is initially placed near the middle of the playing field.

Java classes were constructed for each of these robots implementing the emotion and behavior generating algorithms described in that paper, with the exception that Butler (the waiter) was also given a "go back" behavior in which, after being replenished with treats by Leguin, Butler would go back to its original starting point before resuming its random walk through the playing field, which

represented the reception hall. Except for this difference, the simulated Butler employed the same state variables as the actual robot, and similarly for Leguin

For testing purposes, the test employed a playing field of 50 feet by 50 feet, with 100 moving obstacles (representing the guests in the reception hall). Butler was initialized with 30 treats, and was replenished at each exchange event to that amount. Treat consumption was approximated using an exponential distribution, which was adjusted so that treats would be consumed quickly, to facilitate the speed of observed affective and behavioral state changes.

Observations of the behaviors of these simulated robots in this experimental configuration replicated faithfully the key features of the robots' gross behaviors. In some trials, where Leguin's path was blocked by a "crowd" of guests, Butler grew frustrated and changed to its "intercept" behavior where, instead of executing a random walk to serve guests, Butler moved directly towards Leguin to hasten replenishment. And, whether or not Butler was in intercept mode, when the two robots came within four feet of each other, they entered exchange mode, after which Leguin stopped and waited for Butler to issue the "go home" command, and Butler did so and proceeded with her "go back" behavior. When Leguin returned to the food table, she properly stopped and entered a wait state, waiting for Butler's next refill or hurry command. And when Butler reached her initial position, she properly resumed her treat-serving random walk. Sometimes, as in "real life" her treats were again exhausted before Butler regained her original starting point.

Additional experiments were conducted employing a third robot, named "Vonda", which was essentially a clone of Leguin with the exception of the change of name. Following the above described procedure for configuring the simulation, the Vonda class was compiled and her name added to the configuration file. Running CARS against this configuration file produced a simulation display that showed the displays of all three participating robots. Experimenting with this configuration, CARS ran essentially the same as with only Butler and Leguin, for although Vonda could be placed on the playing field and no objects were permitted to collide with her, she did not change state, nor move, since she did not receive any command to refill or hurry from Butler, which was to be expected. Modifying Butler's code to send refill and hurry commands to Vonda instead of to Leguin produced the expected results – Vonda came to Butler's rescue, while Leguin remained immobile.

Limitations and Extensions

CARS was designed to simulate robotic agent and environmental interactions that have a spatial component which can be tracked over time, such as Murphy's hors d'oeuvres serving robots. Accordingly, CARS is not well suited to applications that do not have such a spatial component. However, many interesting cooperative robotic applications do have spatial components, so that the domain of application for CARS is rather large.

The current CARS design implements only random external (moving) agents in the environment. A future extension to CARS could include optional modules for "smarter" external agents, employing the same approach as that currently used for the cooperating robotic agents. Alternatively, such augmented external agents could be driven by scenario data files generated by other simulations or possibly log data generated during some type of "live" hardware implementation.

Further development of CARS is anticipated to include extensions along these lines.

Conclusions

Developing CARS has demonstrated that simulation techniques can indeed be applied effectively to the domain of affective computing. CARS was able to implement easily the affective control algorithms that were designed for Murphy's robots. And the ease with which robotic modules could be added or changed demonstrated that CARS can be used effectively to prototype and refine such algorithms. Such prototyping can be expected to hasten the development of high-performance affective control algorithms, and to save time and resources in doing so.

Acknowledgments

The authors wish to thank ONR for partial support for this research.

References

[1] Defense Modeling and Simulation Office, Modeling and Simulation Information Analysis Center, A Modeling and Simulation Primer. http://www.education.dmso.mil/ms_primer.asp?a=s4&b=view&c1=272.

[2] Carbonell. Towards a Process Model of Personality Traits. *Artificial Intelligence*, 15:49-74, 1980.

[3] Castelfranchi, C. De Rosis, F. and Falcone, R. Social Attitudes and Personalities in Agents. *In Proceedings of the AAAI Fall Symposium Series on Socially Intelligent Agents*, Menlo Park, CA: AAAI Press, 1997.

[4] Crevier, D., *AI: The Tumultuous History of the Search for Artificial Intelligence*, Basic Books, 1993.

[5] Frijda, N. 1986. *The Emotions*. New-York: Cambridge University Press.

[6] Hayes-Roth, B., Ball, G. and Lisetti, C. and Picard. R. and Stern, A. Panel of Emotions in the Interface. In *Proc. of Intelligent User Interface International Conference,* ACM Press, 1998.

[7] Leventhal, H. and Scherer, K. The Relationship of Emotion to Cognition: A Functional Approach to a Semantic Controversy. *Cognition and Emotion*, 1(1) 3-28, 1987.

[8] Lisetti, C., Personality, Affect, and Emotion Taxonomy for Socially Intelligent Agents, in *Proceedings of FLAIRS 2002*, AAAI Press, 2002.

[9] Lisetti, C. and Gmytrasiewicz, P., Can a Rational Agent Affort to be Affectless? A Formal Approach, *Applied Artificial Intelligence,* Vol. 16:577-609, 2002.

[10] Lisetti, C. and Bianchi, N. Modeling Multimodal Expression of User's Affective Subjective Experience. *User Modeling and User-Adapted Interaction*, 12(1):49-84, 2002.

[11] Murphy, R., Lisetti, C, Irish, L., Tardif, R. and Gage, A., Emotion-Based Control of Cooperating Heterogeneous Mobile Robots, *IEEE Transactions on Robotics and Automation,* Vol. 18, October 2002.

[12] Moffat, D. Personality Parameters and Programs. In *LNCS: Creating Personalities for Synthetic Actors*, Trappl (Ed.), Springer, 1997.

[13] Picard, R., *Affective Computing*, MIT Press, 1997.

[14] Ortony, A., Clore, G. and Collins. *The Cognitive Structure of Emotions*. New-York: Cambridge University Press, 1988.

[15] Wierzbicka, A. Defining Emotion Concepts. *Cognitive Science* 16: 539-581, 1992.