An Agent Architecture for Large-scale Security Simulation

USC/ISI Technical Report GameInt-TR-001

Aaron Botelho, Nima Davarpanah, Jerry Lin, David Mozzacco,
Joseph E. Sutton, Marc Spraragen,
Jim Blythe and Mike Zyda

December 29, 2010

1 Introduction

Human error is perhaps the most prevalent source of vulnerability in a secure system, yet it is often overlooked when testing security. One reason is the difficulty of creating repeatable experiments that introduce human error into large-scale networks. We report on progress on a project to provide this capability using software agents that interact with a networked system under test, operate standard software, follow workflows and introduce human-like errors resulting from models of physiology, emotion and bounded rationality. In an earlier paper we motivated this approach and outlined an architecture based on the Beliefs-Desires-Intentions (BDI) agent framework [Bratman1987, Lin et al.2010]. In this report we describe an initial implementation of the architecture and its application to an example scenario. We then describe future plans.

The remainder of this section outlines the benefits of this approach from a security point of view, the ways in which humans are irrational that can be predicted and modeled, and our overall approach. Section 2 outlines the agent architecture and the scenario we follow, and describes the different agents that interact within this scenario and their implementation. In Section 3 we describe the networking infrastructure that ties the agents together and allows them to communicate while operating virtual machines in which they run everyday email and spreadsheet programs. One aspect underlying the agent's behavior is the role of emotion and related affect, and our approach is described in section 4. We then describe an initial visualization tool in section 5 that allows us to view the overall behavior of a group of agents collaborating on a task as well as investigate the historical behavior of individual agents and their corresponding affectual states. In section 6 we describe a set of experiments that confirm that our

agent system behaves reasonably under reasonable assumptions. We conclude with a discussion of lessons learned and future work.

1.1 Security and human factors

In the literature, computer security is often considered in theoretical terms, for example it may be proven that a certain cryptographic protocol cannot be broken by a plain text attack. In the real world, however, the security of a system comes about through a combination of its capabilities, the software and hardware designed to protect it, and the actions of the users of the system. If that protocol is too complicated or too slow to use, it may be implemented incorrectly or simply disabled when a group is most busy, a time when a cyber attack may be more likely.

Several groups have investigated the effects of usability on system security, *e.g.* [Cranor and Garfinkel2005]. Models of the spread of computer infections also need to take into account that communications between hosts is typically not random, but follows patterns dictated by collaborative tasks or communities of interest. Natural human rhythms associated with sleeping, eating and work hours also affect the patterns of communication and the probabilities of vulnerability to attack. In creating computational models of human factors to facilitate testing, we must consider the properties of groups and also individuals. Below, we briefly discuss our approach to modeling groups. We also consider three broad aspects of individual human behavior: bounded rationality, physiology and emotion.

1.2 Collaborative work

In many of the networks whose security is of interest, human users are engaged in collaborative work, typically geographically distributed. The structure both of the collaborative task and of the organization performing it determines the nature and patterns of network traffic to a large extent and therefore should be modeled when testing security approaches. In the scenario that we describe below, a small team is engaged in a report generation task that involves gathering information from various sources on the internet and populating a shared spreadsheet that is implemented in a cloud system. The team has one central manager and one agent designated as IT support as well as a number of information workers. In a related experiment, the overall task is driven by the need to respond to a natural disaster, leading to an externally generated burst of activity for the group.

Different kinds of groups have different organizations in order to perform their tasks. Some are more loosely or democratically organized than the central hub-and-spoke structure we have considered here, and in some, organizational patterns might emerge temporarily in response to the environment. We are interested in modeling a number of different typical organizational structures, focusing on their impacts on system security.

1.3 Physiology and emotion

Humans are not machines and are strongly influenced by physiology and emotion in both decision-making and performance of lower-level tasks. There has been a great deal of work understanding how fatigue builds in humans and its effect on performance in a variety of settings. Our agents include simple models of fatigue inspired by some of this work. We focus in particular on time-on-task rather than the effects of sleep deprivation, since our intended experiments do not yet span timescales that make this relevant.

Section 4.2 describes our approach to modeling emotion and its effect on human decision-making in more detail. These are inspired by work on computational models of emotion including appraisal theory, *e.g.* [Ortony, Clore, and Collins1988] and associative networks with spreading activation [Anderson1983]. However, the approach described there is not implemented in our current agent system, but is intended for a future version.

1.4 Bounded rationality

Many agent systems model cognition in terms of rational choice, *i.e.* the agents attach a utility to probability distributions of possible outcomes of their actions and seek to maximize their expected utility. It is well-known, however, that humans do not behave in this way [Tversky and Kahneman1981]. Instead, they apply heuristic decision-making that, while provably sub-optimal, allows them to make decisions when the full space of actions cannot be explored and the full utility function is probably unknown. Human decision-making behavior cannot only be explained by limited information or processing time, however, but also by certain systematic differences from utility theory. Humans tend to discount the value of future gains or losses compared with current ones even when the overall value function is known not to be time-dependent. Humans are more conservative about risk when considering gains than they are when considering losses, however the distinction between the two is often based on a heuristic and arbitrary 'zero' point in the space of possible rewards for actions. Humans also make systematic errors in judgments of probabilities that effect their decisions under uncertainty, for example believing that more easily remembered events are more common. Once humans settle on a decision, they tend to put more effort in seeking confirmatory rather than disconfirmatory evidence.

Our current agent framework does not model bounded rationality, though it has been argued to play an important role in security decisions such as neglecting to install security software based on cost or convenience or ignoring warnings. We are investigating ways to include some of these effects in future versions of our agents.

2 Agents

For our scenario we use autonomous agents that collaborate on tasks to meet their top level goal. We created specialized agents that handle different tasks in the scenario. Attributes given to the individual agents are used to create a base level of fidelity and

to allow for diversity in the simulations. The core of each agent is based on a standard belief, desires and intentions (BDI) architecture[Morley and Myers2004]. Each agent is initialized with goals, beliefs, intentions, and a cognitive state. We will discuss the extensions to the BDI system in a later section. The following are descriptions of the attributes and actions of each agent in our scenario.

2.1 Agents and Organizations

Traffic patterns and information flow are very important to the security of a networked system. The level of activity in a network depends on daily and weekly work schedules, and so on time of day and particular deadlines as individuals perform tasks. If a node is infected, the probability of neighboring nodes becoming infected may depend on the number of emails sent between individuals operating the nodes or patterns of usage of networked drives. Our scenarios therefore model organizations comprising teams of agents with shared as well as individual goals. Network traffic arises as agents work together to perform collaborative tasks and therefore follows a pattern that is more similar to that in human organizations. Agents in the organization have a number of differentiated roles, including managers, workers and IT support. However, the agents share a common understanding of their world in terms of a means to communicate and shared goals. In the next sections we describe the shared agent structure and then the features of the specialized agent types.

2.2 Agent Base

The Agent Base is a generalized agent that contains the attributes and actions that are fundamental to all agents in our scenario. Each specialized agent in the scenario is a subclass that extends the capabilities and beliefs of the Agent Base.

In the current implementation, on SPARK, agents use an easily extensible predicate representation scheme in which new attributes can be added as needed for new scenarios. Below we describe the knowledge used for the example scenario that we anticipate will be relevant for other scenarios under construction. We draw a distinction between *attributes*, which are predicates that represent features of the agent that are possibly unknown and beyond its control, such as tiredness, and *beliefs*, covering predicates that represent the agents beliefs, such as its position in the organization. These are not all classifiable as "knowledge" since the agent's beliefs about the environment may be incorrect but still are used to guide its actions.

2.2.1 Attributes

All agents possess a number of attributes that control their physiology and aspects of their emotions. This includes the agent's current level of tiredness and also its tireability, or how quickly it becomes tired as it performs tasks and with the passage of time.

2.2.2 Beliefs

All agents have knowledge of their job position, which determines the type of specialization and role of the agent in the scenario. Each agent is also an assigned work identification number to uniquely identify them within an agents social network and the job hierarchy. Along with the worker identification number the Agent Base contains knowledge corresponding to the agent's computer identification number and computer applications, as well as the current number of tasks that the agent must complete.

2.2.3 Actions

The Agent Base contains an initialization cue that triggers each agent to set up its specialized state. After initializing its agent state, the Agent Base proceeds to work on its current goals. While the agent is taking actions to meet its goal, it listens for communication from othe agents. The Agent Base defines the actions used to communicate, which include receiving and initiating telephone messages, face-to-face communication and email messages.

2.3 Agent IT

The IT agent is a collaborative agent that works in Information Technology Department in the example scenario. With the help of an IT agent, other agents can respond better to unforeseen circumstances. Its duty is to help other agents with errors, bugs and viruses the agent's computer might have. Figure 1 illustrates the main work flow of the IT agent.

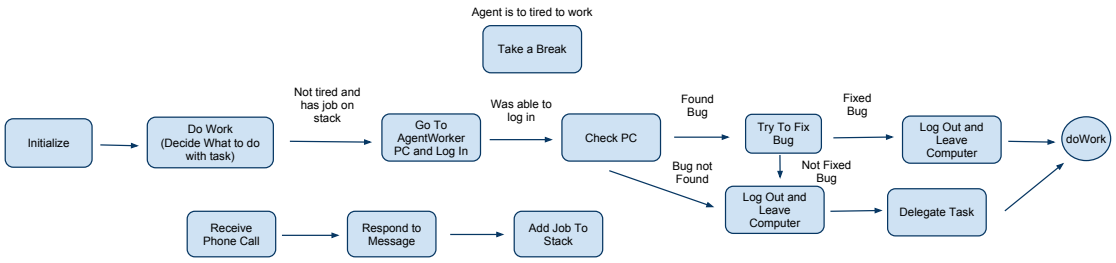

Figure 1: The main work flow of the IT agent

2.3.1 Beliefs and Attributes

The IT agent knowledge base contains information about the different types of errors, bugs and viruses that can go wrong while an agent is working. It also has attributes that define its technological competency and the thresholds needed to diagnose and fix problems.

2.3.2 Actions

The IT agent's goal is to keep software in working condition. It can log in to another agent's computer to perform the diagnosis on the computer. If it finds a bug it can attempt to fix it. If the IT agent fails to fix the computer it can delegate the tasks to another IT agent within its social network. If the computer was fixed the IT agent logs out and returns to its office where it waits for messages from agents asking for help.

2.4 Agent Worker

The Worker agent's goal is to fill the spreadsheet cells that were assigned to it by its corresponding Manager. The Worker agent searches in online websites for the information needed to fill the spreadsheet cells. A Worker can also work collaboratively with its co-workers to meet its goal, by delegating tasks to them. Figure 2 illustrates the main work flow of the Worker agent.

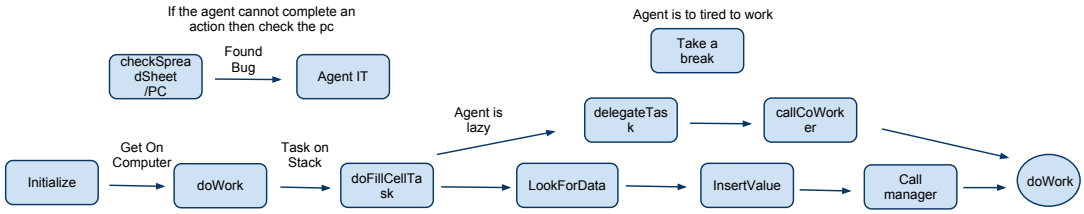

Figure 2: The main work flow of the Worker

2.4.1 Beliefs and Attributes

The Worker agent knowledge base contains information about websites containing data to insert into the spreadsheet. It also keeps a track of what values it has inserted into the spreadsheet it was assigned. The Worker agent also has the attribute related to the tendency of saving the correct value into the spreadsheet and the likelihood of delegating a task.

2.4.2 Actions

The Worker agent begins the scenario by logging in to its assigned computer. It then waits for a call from the Manager agent or works on any tasks it currently has. If the Worker agent has tasks to fill the spreadsheet it will begin to look for data online. Then it will fill the assign spreadsheet cell with the information it has retrieved. If the Worker has too many tasks to handle it will attempt to delegate one of the tasks to a co-worker. Periodically, the Worker agent will validate the information on the spreadsheet. If the information is compromised and it cannot fix the problem itself, it will contact an IT

to help with the problem. After the Worker agent has finished a task it will report the completed task to Manager agent.

2.5 Agent Manager

The Manager agent is derived from the Worker agent and therefore contains all of its emotions and capabilities. Unlike the Worker agent, the Manager agent is able to accept large tasks from the Scenario Director and break them into smaller tasks to be delegated to other Manager or Worker agents under there respective managment. Finally the Manager agent is able to verify or delegate verification tasks to Worker agents to check if spreadsheets was properly filled in.

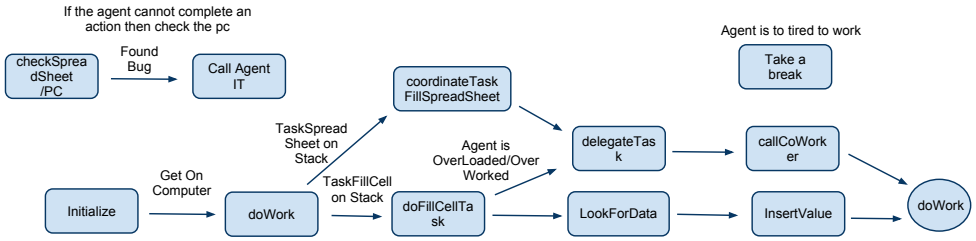

Figure 3: The main work flow of the Manager

2.5.1 Beliefs and Attributes

The Manager agent's knowledge base contains information about its given tasks and its computer. The task it is given when it is initialized is to fill a spreadsheet of certain dimensions. It will then divide the spreadsheet tasks into subtasks and keep record of the number of subs tasks it has created. The attributes it has includes the tendency to look for data, check its computer and tendency to delegate tasks.

2.5.2 Actions

In the present implementation the Worker agent will call its Manager agent to notify that the given task assigned was completed. The Manager agent assumes the Worker agent has completed the task correctly. The Manager agent currently communicates with one or more of each type of the following agents: Scenario Director, Worker and IT.

The Manager agent is initiated by the Scenario Director. When the Scenario Director calls the Manager agent's Knowledge Base is updated with the details and size of the task. The manager assigns subtasks individually to worker agents in its group. The Manager additionally waits until the worker calls him to notify that the given subtask has been completed. Finally when all subtasks have been completed the Manager

agent's Knowledge Base is updated. The Manager agent then waits for more tasks to be assigned by the Scenario Director, possibly waits indefinitely until the end of the simulation.

As a derived Worker agent the Manager agent can have problems with his computer throughout the process mentioned above. The Manager agent can attempt to fix its computer and calls the IT if the computer is infected. Thus delays can occur throughout the process. For a full overview of these capabilities please refer to the Agent Worker section on page 6.

2.6 Scenario Director

The main purpose of the Scenario Director is to ensure that any pre-determined events within a scenario unfold as expected. It is implemented as an agent in order to share the agent framework for communication, but it does not model other agent characteristics such as tireability. The Scenario Director monitors the scenario and directs events to occur according to a predefined script. The agents in the scenario are unaware of the Scenario Director and its influence. Figure 4 illustrates the main work flow of the Scenario Director.

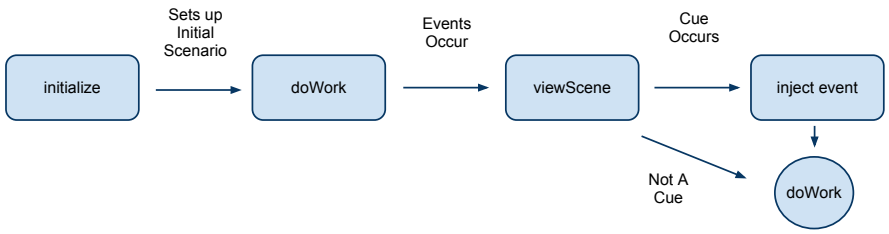

Figure 4: The main work flow of the Scenario Director

2.6.1 Beliefs and Attributes

The Scenario Director's knowledge base contains information about events that will cue the director to cause scripted events to occur. Along with information on all of the agents involved in the scenario.

2.6.2 Actions

The Scenario Director is able to see all events that occur amongst all agents by way of the Subscription Service (see Subscription Service sub section in the Infrastructure

section on page 9 for details). The Scenario Director responds according to the cues it is waiting for. In the case of the given scenario the Scenario Director is waiting for a certain spreadsheet cell to be filled in order to trigger a bug/attack on the spreadsheet.

3 Infrastructure

The infrastructure consists of multiple processes passing messages on dedicated ports for each process. Processes may exist on the same machine or multiple machines connected on a network. Figure 5 is an overview of the process connections. The Process Controller spawns the Middleware, Logger, Visualizer, Agents, and Scenario Director processes, depicted by the dashed line ending with a solid square. Each process listens on one or more ports for incoming messages, the messages use a general format throughout the network, to be processed by the receiving process. Message communication links are represented by a solid line with a solid arrow pointing to the receiving process. The Middleware process uses inner communication between sub processes (threads), illistrated with a dotted line and solid arrows. The Logger and Visualizer both access the centeral DataBase Managment System (DBMS) to access and record the message history, depicted with a solid line and a solid circle connecting to the DBMS.

There are six different types of processes on the network, with not all existing on each machine; Processes Controller, one instance per machine, all machines in the network. Middleware, at most one instance per machine, but at least one per network. Subscribers (e.g. Loggers), zero or more instances per machine, but not the same type of Subscriber per machine. Agent(s), zero or more instances per machine, each instance network wide having a unique ID to distinguish the Agents (to route a message approperately). Scenario Director, exactly one instance per network. Commander, exactly one instance per network at a static address.

3.1 Agent Api

The agents use an API to communicate with other agents and broadcast their state information to the subscription service. The agents are defined using an agent platform called SPARK [Morley and Myers2004]. SPARK is written in Python and executed in Java using a package called Jython. The agent API itself has Spark/Python wrappers to the interface that sits at the highest virtual level of the execution stack (figure 6), Java. Using Java for the API posed many advantages over writing the API in pure python (Jython); All Python code is executed in Java. Java's object-oriented style made it easy to reuse classes for the Middleware. Serializable java objects were used to pass complex data forms between communicating agents. Finally, Java eased the debugging and unit testing process for the agents and Middleware.

When one agent communicates with another, the API abstracts the communication layer to several API calls based on the type of communication (*e.g.* in person, phone, email). The API wraps the communication message in a generic message that is passed to a Middleware server (see Middleware sub-section below for details). The Middleware server routes the message to the appropriate agent. The receving agent unwrapps

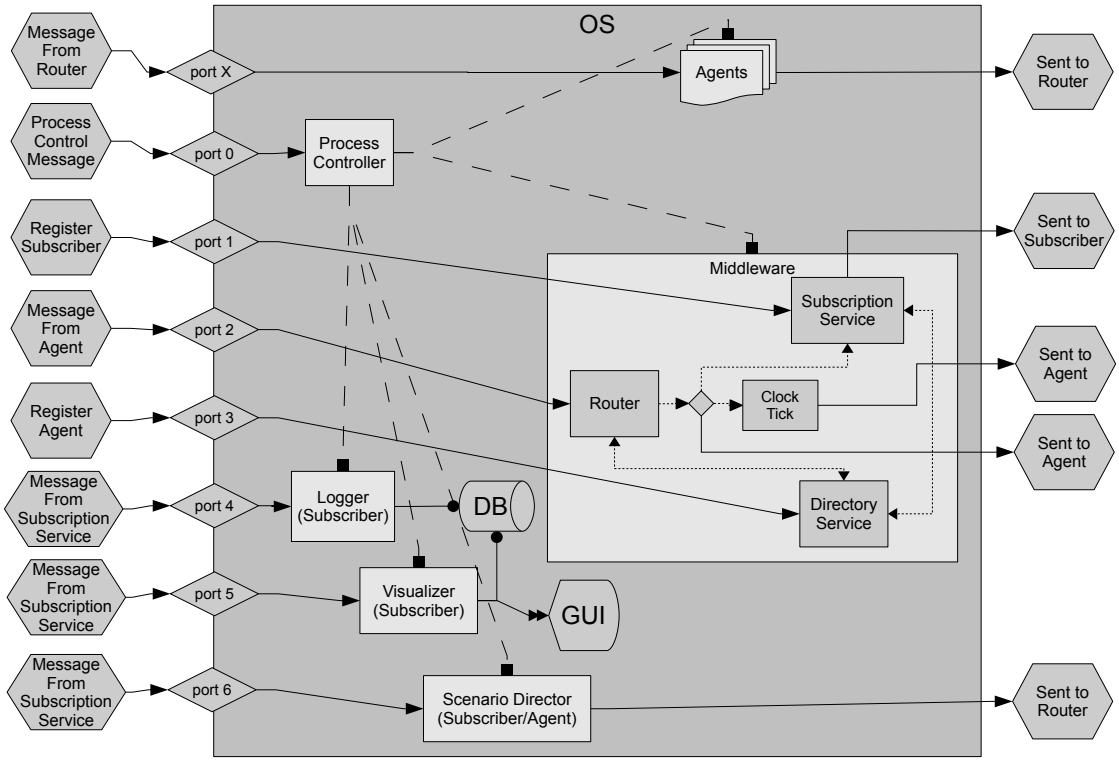

Figure 5: Infrastructure

the message and puts it in a queue to be processed later.

3.2 Middleware

The Middleware process consists of three subprocesses running on threads: the router, directory service and subscription service.

3.2.1 Router

Upon receiving a new message from an Agent the Router looks up the intended recipient in the Directory service, to translate the Agent ID to a network address. A clock tick message is sent to all Agents, to preserve ordering in dependent messages([Lamport1978]). A copy of the received message is added to the Subscription Service Dispatch queue and finally the message is sent directly to the recipient agent's network address on their unique port. See figure 7.

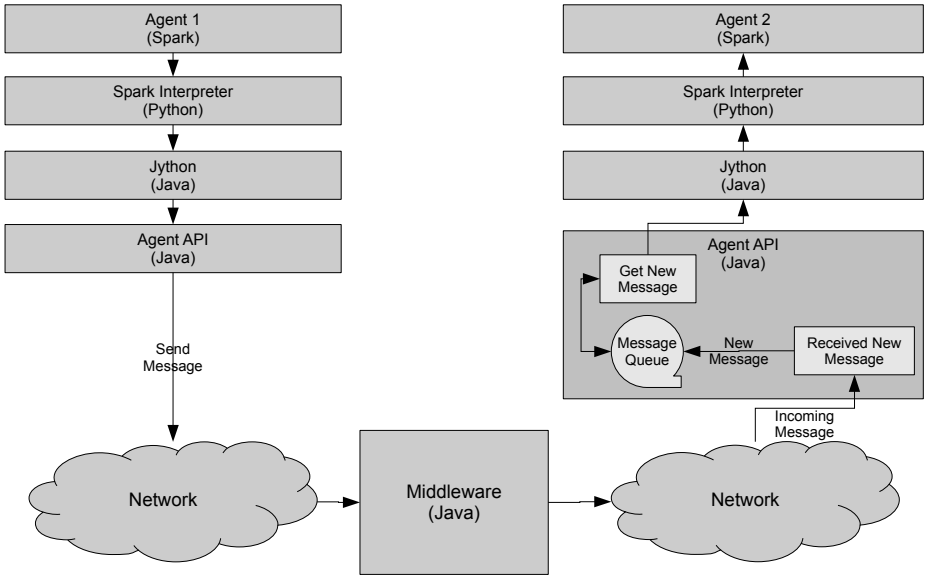

Figure 6: Execution Stack

3.2.2 Directory Service

The Directory Service maintains a table of all the known agents, mapping their unique IDs to their network address. The Agent when instantiated sends a registration message to a known Directory Service, thereby adding the Agent to its lookup table (figure 8). The registration and lookup table is based on the idea of a DNS system(RFC-882, RFC-883).

3.2.3 Subscription Service

The Subscription Service maintains a list of all known subscribers, for example the Logger, Visualizer and Scenerio Director. The Subscription Service also dispatches messages to all subscribers in its list. When a message is added to the dispatch queue by the Router, the Router releases a semaphore, that the dispatch thread is waiting on. When the dispatch thread is released it reads then removes the top item in the dispatch queue, sends the message to all subscribers in the subscriber list and then goes back to waiting on the semaphore. See figure 7.

3.3 Subscribers

The subscribers read all messages sent over the network. They do not send messages back to any agent, with the exception of one hybrid process, the Scenario Director.

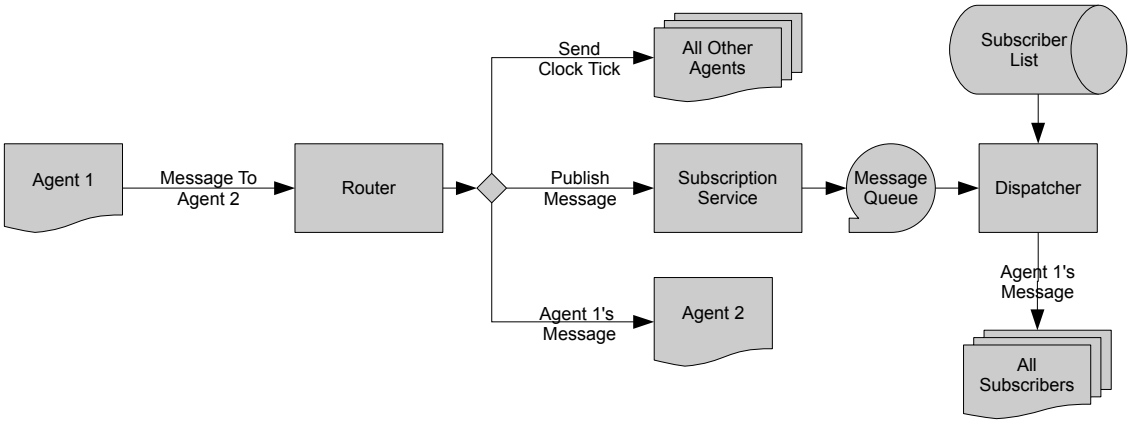

Figure 7: Agent Message

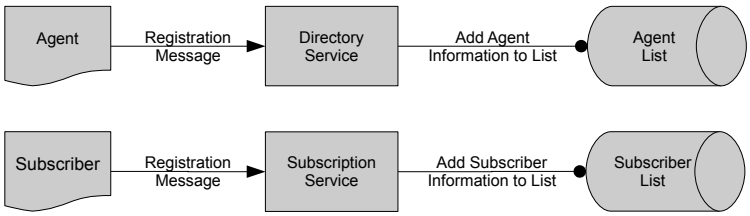

Figure 8: Registration

3.3.1 Logger

The Logger records all messages sent from agents, storing them in a DBMS. The recorded data is used to evaluate the behavior of the Agents, and replay Scenarios.

3.3.2 Visualizer

The Visualizer is a Graphic User Interface (GUI) to the Scenario, used both for live view or re-playing recorded data from the DBMS. For more details see the Visualizer section on page 24.

3.3.3 Scenario Director

Unlike the other subscribers, the Scenario Director sees all messages and interacts with the agents, pushing some agents to behave in a particular way to keep the scenario on track. See the Scenario Director sub-section on page 8, for details.

3.4 Process Commander & Controller

Process control is very important in a large distributed system. The order in which processes startup and exit is also important. For instance, the Middleware service has to start up before any agent, so the agent can register its self with the Directory Services. The same goes for the Subscribers(ex. Logger). The Middleware must be running prior, otherwise the Logger can not register with the Subscription Service.

3.4.1 Spawning Processes

The Process Controller allows the overall infrastructure to be dynamic and scalable. Each process can either be started alone or via the Processes Controller, which waits for messages from the Process Commander or Scenario Director (figure 9). The Scenario Director sends a message to the Controller to start up the Agents after the Scenario Director has created each agent's knowledge base file. Therefore, the Controller process needs to be already running in order for the Scenario Director to spawn the agents.

3.4.2 Process Order

The Process Commander enforces an order to in the process startup, waiting for each process to confirm it is running before the issuing the command to the Controller to startup the next process. The order in which our system starts the processes using the Commander is as follows:

1. Controller
2. Middleware
3. Subscribers(Logger, Visualizer)
4. Scenario Director
5. Agent(s)

3.4.3 Killing Processes

The Process Controller also has the ability to send a special termination message to all the processes. When a process receives this message it sends back a conformation message, then proceeds to exit gracefully. The order in which the processes die is also important. For example, if the Middleware dies before the Agents do it cannot send the termination messages to the agents. This is easily solved by sending the termination messages in reverse order to the start process order, waiting for each process to confirm it received the message and is exiting gracefully.

3.5 Scaling for the Future

Multi-machine Clock Tick (see Router in Middleware sub-section for details) synchronization, Directory lookup, and Subscription Service posses the major issues in scaling to thousands or tens of thousands of agents running concurrently on a single network.

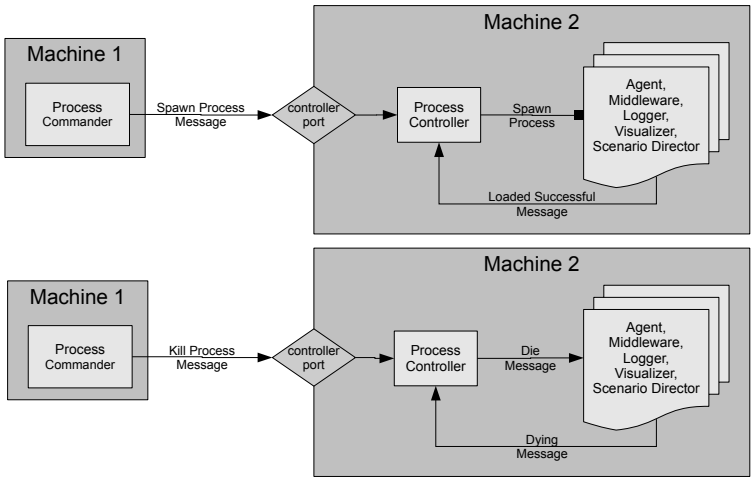

Figure 9: Process Spawning

The Clock Ticks will need to be syncronized between Routers on all machines running the Middleware. This is done using a list of all known Routers on each Middleware machine. The alternative is to use a central server that maintains a list Middleware servers and sends the Clock Ticks to them. The Directory Service would need a root server to maintain a list of all agents. When a Directory Service does not have an agent in its list, it will ask the root server for the mapping and then cache it locally (figure 10). The subscription will need to be changed to use a root Subscription Server that a subscriber will register with. The root Subscription Server will then register the subscriber to all known Subscription Services on the network.

4 Emotions and Appraisal Theory

4.1 Introduction

Research on the interaction between emotion and cognition has become particularly active in the last twenty-five years. Notably, the work by Bechara and Damasio [Bechara, Damasio, and Damasio2000] showed the necessity of emotion for decision making: loss of emotion likely leads to indecision or disadvantageous life decisions. This result challenged and largely overthrew the classical view that emotions could only cloud rationality, though that effect has also been documented [Gmytrasiewicz and Lisetti2000].

Also motivating research on emotion is the characterization of emotion as an interrupt alarm signal to cognition [Simon1967, Bower1992]. The signal is particularly responsible for heightening the importance of concepts associated with the emotional episode, and for refocusing attention [Ohman, Flykt, and Esteves2001, Bower1992]

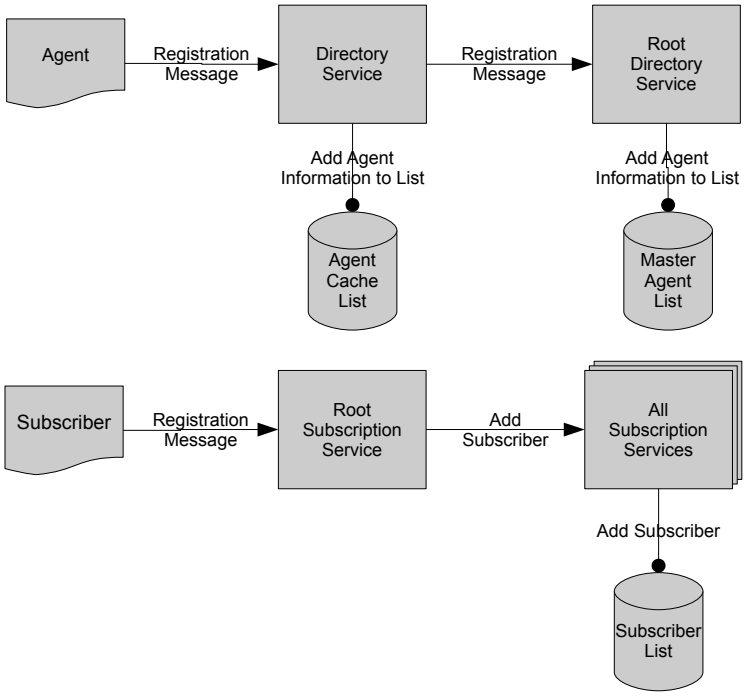

Figure 10: New Registration Process

(causing distraction from a non-emotionally relevant task at hand when an emotional episode occurs). Damasio also asserted that emotion facilitates special recall of concepts when high emotional arousal occurs [Damasio and Sutherland1994].

We believe that seemingly disparate emotional theories and experimental results can be integrated smoothly into a single computational model of human cognition. As part of the rise of emotion research in the AI and cognitive science communities, researchers have created several computational models of cognition and emotion, based on psychological theories and experimentation. A typical implementation of emotion generation is bound to a single theory, which usually conflicts with other theories on which factors generate emotion and how. Computational models of affect tend to focus on a single effect of emotion on cognition. These research practices have led to incomplete, competing models which leave aside the question of a complete integration of emotion and cognition. We set forth proposals for a deeper integration than previous cognitive-emotional architectures, and present the design of a cognitive architecture, EmoCog, which embodies these ideas.

4.1.1 Background

Our approach is fundamentally built on theoretical and experimental work in psychology, cognitive systems, and neuroscience. Appraisal theories dominate the realm of computational models of emotion generation. Appraisal theory generally argues that people are constantly evaluating their environment, and that evaluations result in emotions such as fear or anger. Traditional game playing programs which evaluate their environment and/or self are not emotional, since they do not produce the necessary appraisal data for emotion and affect. There are many different appraisal theories, notably those of OCC [Ortony, Clore, and Collins1988], Frijda [Frijda1987], Smith and Lazarus [Smith and Lazarus1990], and Scherer [Scherer2001]. Each theory differs in its appraisal variables and the manner in which appraisals are generated (e.g. simultaneously vs. specific order).

Several theories inform our work on emotional cognitive effects. The Somatic Marker Theory predicts that emotionally enhanced memory is useful for decision-making, as shown in the Iowa Gambling Task [Bechara, Damasio, and Damasio2000]. According to similar experiments, "gut feelings" during emotionally stressful moments are a heuristic to making a decision quickly, bypassing cognitive evaluation [Slovic et al.2007, Finucane et al.2000]. The related mood congruence theory [Bower1983, Bower1992] hypothesizes that facts or concepts learned during a positive or negative mood are thereafter easier to remember when in a similar mood. Conversely, the Yerkes-Dodson law [Yerkes and Dodson1908] predicted that high levels of emotional arousal creates distraction from non-emotionally relevant tasks at hand [Kaufman1999]. The cue utilization theory [Easterbrook1959] elaborates this effect: under high levels of arousal, environmental or internal cues not central to the arousing agent or situation will be increasingly ignored.

Simon's emotion-as-interrupt theory [Simon1967] highlights autonomic arousal as a factor of emotion. Many of the widely cited emotion generation theories use arousal as a factor and can be applied to our model. Emotion generation theories usually also incorporate valence (degree of pleasantness or unpleasantness), which we can use to model further emotional effects on cognition, such as mood-dependent retrieval.

We also draw from a long tradition of work in computational cognitive architectures. Such systems usually try to address cognition as a whole. Our work has been most directly influenced by ACT-R [Anderson et al.2004], CLARION [Sun2006], PRS [Ingrand, Georgeff, and Rao1992], and Soar [Laird2008]. See [Langley, Laird, and Rogers2009] for discussion on this topic.

4.1.2 Related Work

Integration of emotions into cognitive architecture can be broken down into two separate parts:

1. Emotion generation - how cognitive processes play in the generation and decay of emotions

2. Emotional affect - how emotional signals, once generated, effect cognitive processes such as learning or planning

Some researchers have theoretically integrated emotion and cognition [Schorr2001, Bower1992] but leave out many details about the processes and the data that underlie them. Several computational models have been developed in attempt to flesh out some of these details.

The prominent systems that address emotion generation in a cognitive architecture include Soar-Emote [Marinier, Laird, and Lewis2009], EMA [Marsella and Gratch2009], and WASABI [Becker-Asano and Wachsmuth2010]. The Soar-Emote work discusses how appraisal would occur in Soar, using Newell's theory of cognitive control. It is bound to a number of theoretical assumptions that stem from a single theory of emotion. EMA addresses the process of appraisal over a previously generated plan. Neither Soar-Emote nor EMA address how various cognitive processes would influence appraisal. WASABI is closest to our work on emotion generation. It presents primary and secondary emotions, where secondary emotions depend on past experiences and learned expectations and map to three discrete emotions (hope, fear, relief). The scope of this work lacks interaction with most cognitive processes and is limited to explaining few emotions.

Prominent systems that address effects of emotion on cognition include Soar-Emote, EMA, ACT-R [Cochran, Lee, and Chown2006, Fum and Stocco2004], and MAMID. Soar-Emote has work limited to how emotion may be input to reinforcement learning [Marinier and Laird2008]. EMA models generation of coping strategies following an emotional episode (e.g. change own beliefs). What are still missing are mechanisms for how the cognitive processes can be affected. Cochran's work in is limited to how emotional arousal may affect memory and Fum's work is similarly limited to how emotional memory would affect recall and subsequently decision making. MAMID's emotional effects on cognition are limited to altering the speed and parameters of a prescribed perception-action cognitive cycle. No previous computational model has attempted to integrate all of this work and other emotional effects on the function of cognitive processes in a single cognitive architecture or under a single theoretical perspective.

4.1.3 Overview

The remainder of this section presents our propositions. This can be broken down into three parts:

1. EmoCog Architecture - modules, interactions, and data structures required
2. Mechanisms - processes that operate within EmoCog in context of emotion generation and affect
3. Discussion and Examples - rational and alternate perspectives on EmoCog's design, and some examples to illustrate ability to model observed phenomenon

4.2 Approach

The primary theoretical proposals for our computational model of emotion and cognition require certain programmatical groundwork to implement in a cognitive architecture. We outline the key design decisions of EmoCog, but leave detailed discussion

of implementation to a future report. The novel features of EmoCog are the interactions between emotion and "rational" cognitive processes. In this particular version of our proposed architecture, we focus on emotion generation and emotional effect on memory, attention, and planning.

4.3 Architecture

The architecture diagram is shown in figure 11. At a high level, the architecture bears much resemblance to existing cognitive architectures such as Soar, CLARION, EPIC, and ACT-R. The potential cognitive modules are not limited to those shown.

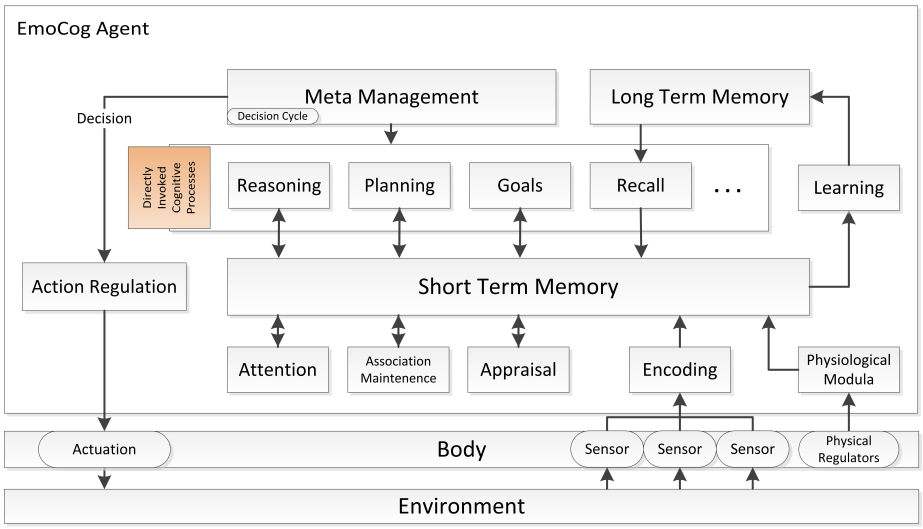

Figure 11: EmoCog cognitive architecture

EmoCog's short-term memory is based on ideas outlined by Bower in his associative network theory of emotions [Bower1981], and the spreading activation theory of memory by Anderson [Anderson1983], similar to that which has been implemented in ACT-R [Anderson et al.2004]. EmoCog's model of memory (both long-term and short-term) is a graph made up of generic nodes and links, and will function as an associative and semantic network.

There are several types of links between nodes, each with a label, a value, start node, end node, and optional direction. All nodes that are connected have an association link, which carries an association strength value. Associative link creation, reinforcement, and decay are all managed by the association management module (see below). In addition to associative links, there can be semantic links between nodes (e.g., causality), which can also carry values. These semantic links are maintained by cognitive processes (e.g. causal inference placing a causal link).

Each node can represent, but is not necessarily limited to, an episode, object, deadline, utility, concept, plan step, or procedure. The following node features are used by the appraisal system:

1. **Current arousal** (range 0 to 1): emotional arousal at current time
2. **Remembered arousal** (range 0 to 1): average arousal over time
3. **Current valence** (range -1 to 1): degree of like/dislike
4. **Remembered valence** (range -1 to 1): average valence over time

Other node features, such as recency of recall and how many times the node has been brought into working memory before, are not used by appraisal.

The current arousal and valence values are generated by the appraisal module. That process is presented in the following section. Remembered arousal and valence are averages of the current arousal and valence over time, which can span many episodes of the agent's experience. The remembered arousal feature allows modeling the recall facility of nodes associated with strong emotions [Damasio and Sutherland1994]. The inclusion of remembered valence allows modeling mood-state dependent retrieval [Bower1992].

4.4 Mechanisms

The mechanism set of EmoCog may be broken down into three key process groups: directly controlled cognitive processes, automatic cognitive processes, and meta-management. Figure 11 identifies the cognitive modules we propose to be directly controlled through meta-management. All other processes are assumed to be automatic and run in parallel.

For purposes of this section, the details of the majority of these modules are abstracted, as we defer discussion of these to later reports. The sensory and encoding module handles the addition of new nodes into short-term memory from perception. Action regulation can be seen as the cognitive architecture's interface (mainly output) to the body.

The attention module is responsible for selecting an associated cluster of nodes for cognitive elaboration. Selection determines which node cluster to use in cognitive elaboration by finding a single node with the greatest weighted sum of current arousal, associated utility, and associated urgency. All nodes directly and indirectly associated with the core node are also selected, using a breadth first search until a threshold is met to form the cluster. The shifting of attention via emotional processes [Simon1967, Bower1992] has been marginally addressed in architectures such as CogAff [Sloman2001]. Meta-management is able to exercise limited executive control over attention by setting the weight of each of these parameters.

The association maintenance module performs spreading activation to create association links, and to reinforce current associations in working memory. With time and neglect, associations between nodes decay in long term memory. For example, when an object is perceived, a node is created for the instance of object perceived. If the object is in long term memory, an association must be made to the symbol representing that object in long term memory. If the object is previously unknown, associations can be made through various methods (e.g. matching by analogy or temporal relation).

The appraisal module adjusts the current arousal and valence values of nodes in short-term memory. When a node enters short-term memory, association maintenance occurs, and then the node is subjected to immediate first level appraisal. This appraisal is based on remembered arousal and valence and innate feeling (e.g. evolutionary dislike of predator or a negative utility node) if remembered arousal and valence is unavailable. The innate feeling is typically grounded in the body (e.g. pain is bad, and intensity of pain dictates arousal).

The node will be subject to reappraisal for as long as it remains in short-term memory. This may be best characterized as the influence of associated nodes on how an agent feels about the focal node. A graphical walk takes place on associated nodes, propagating the current arousal and valence values (these values are scaled down based on association strength). The traversal is terminated, if not earlier, when all nodes in short-term memory have contributed. Four values are produced by this process: sum of arousal of negative valence associations, sum of arousal of positive valence associations, average negative valence, and average positive valence. The valence with a higher summed arousal will dominate and inhibit opposing valence. The appraisal module then incorporates the average arousal and valence into the node's current arousal and valence. When a new node enters a cluster and is appraised using first level appraisal, it will similarly influence neighboring nodes in an outward fashion.

Overall mood of the agent will also be maintained by the appraisal process. The current intention is to compute mood as an average of all current arousal and valence of nodes across working memory. A single node's appraisal can still influence our mood over a long period of time, given that the node remains in working memory. This needs elaboration, however, as mood is not only an overall emotional state based on working memory, but may persist, decay, or change independent of the changing emotionally charged nodes in working memory.

Physiological signals will relate the needs of the body to the cognitive architecture. In the human body, these signals might be of hunger, thirst, or fatigue. The physiological modula interprets a body signal and maintains a node in short term memory as well as associated urgency and utility. The strength of the signal is directly translated to an interpretation of urgency, while utility is innate.

The meta-management module is where metacognition and cognitive control will take place. The vital components of this module are the metacognitive rules, decision cycle, and list of directly invoked cognitive processes. In practice, the metacognitive rules and the rules describing cognitive processes are represented and applied within the same reasoning platform. Actions, in addition to existing as nodes in the associative memory, are also reasoned about and decomposed within the same platform. This approach gives EmoCog an unprecedented ability to represent interactions between emotional and physiological processes and cognitive processes such as planning and inference.

The decision cycle is the driving force of the meta-management. It typically progresses as follows:

1. Perception - Short term memory is updated with information from perception.
2. Attention - Metacognitive rules determine weights of attention parameters. Attention module is invoked.

3. Elaboration - The node(s) which gain attentional focus are given limited cognitive processing. Rules of the metacognitive module choose which cognitive process runs[1].

4. Decision evaluation - Metacognitive rules determine if enough elaboration has been performed.

5. Action selection - If elaboration has produced a set of candidate actions, one is selected based on metacognitive rules that weigh utility and emotional bias.

6. Action execution - If there is a selected action, it is initiated. The decision cycle is then repeated. Note that subsequent decisions, or exogenous events, may interrupt the execution of the action.

During the elaboration phase, individual cognitive processes are invoked through metacognition, although they share the same rule space. All cognitive processes execute in an anytime fashion, with a limited amount of available computation before the elaboration process repeats, possibly switching attention. Cognitive processes are only able to use the cluster of nodes under attention focus.

4.5 Discussion and Examples

We view EmoCog as an embodiment of principles needed for full integration of emotion and cognitive architecture and it will be particularly apt for modeling affective behavior as described in psychology and neuroscience literature.

One particular phenomenon we address is that of emotions both as interference and heuristic. It was observed that emotional signals can disrupt normal cognitive function, particularly when not relevant to the processing at hand.

For example, an agent is assigned a cognitive task to recall and output a list of words in order from long-term memory, under a deadline. Attention is focused on the first word and the node in associative memory representing this word. The metamanagement invokes the recall process to find the most strongly associated node. After some iteration, several nodes are recalled into short-term memory via this cycle. At some point, the word "tiger" is retrieved and following the next recall cycle, the most strongly associated node of a traumatic "tiger attack" experience is recalled. That node has high activation strength due to high remembered arousal and extreme negative valence. When the "tiger attack" node is brought into working memory, an appraisal based on the remembered arousal and valence is assigned to the node's current arousal and valence. This causes the attention focus to be drawn away from the task to dwell on the tiger attack. Meanwhile, other nodes which do not hold attention focus have their arousal levels decay, allowing the dominance of the aroused thought.

Metamanagement, referencing the agent's goals, attempts to refocus attention to the task by raising the weight of utility. The emotional episode, however, is so strong, that the thought of a tiger attack continues to hold the agent's attention. The attempt to return attention to the task succeeds only when the urgency of the task also increases,

[1]Processes like learning are automatic and are not among those selected

due to impending deadline. These rules in metamanagement are used to reason over the various cognitive tasks. It is similar to Soar's metacognition in this regard.

This particular model of metamanagement stresses the importance of metacognition when our emotions can lead us astray. A person could have been taught to ignore emotionally compelling issues to focus on his work, so he may try to do so, but emotions are very difficult to fully ignore. Sufficient emotional arousal will wrestle cognitive attention away from a rational train of thought. "People who are more rational don't perceive emotion less, they just regulate it better" [De Martino et al.2006].

If a person focuses on a certain task, usually irrelevant emotions fade, but it is not necessary that he has completely forgotten about the invoking fact, it's that it has been tuned out. Neuron signal strength typically decays over time, so under the impression that emotional signals occur in the human brain as simple neurological pulses, we model current arousal of unattended nodes to decay similarly, allowing concentration on a task. That is, unless something particularly compelling draws attention away. There are also well studied mechanisms of signal inhibition and winner take all from neuroscience literature, which we leverage by having the appraisal process inhibit and suppress nodes excluded from the attention cluster.

When relevant to a task, emotion can serve as a heuristic for various types of cognitive processes. Emotion acting on the recall process can model the emotionally-enhanced recall demonstrated in the Iowa Gambling Task, and also model Bower's mood-congruent retrieval effect. For instance, an agent wins a lottery by picking the number 7. The agent creates an association link between a node containing the number 7 and a node containing the experience of winning. The appraisal process confers higher arousal and positive valence to the number 7 via its association with winning. When the prospect of picking a number to win another lottery becomes the agent's goal, 7 is more likely to be recalled than other numbers, as it is positively associated with winning ("lucky 7!"). The agent's mood will also influence the choice. An agent in a positive-valence mood will be more likely to recall 7, as that number has the highest valence among the choices in long-term memory.

Since all cognitive processes work with the associative network and emotional data is embedded within all the nodes, any process can use emotion data to model emotional affect. For example, arousal and congruence may influence the action and goal choices an agent makes when it constructs a plan, and also the fidelity with which it executes a plan. The agent may omit or curtail steps whose actions or objects have lower arousal, even though they are logically necessary to the plan.

EmoCog is designed to be extremely flexible, so that further dimensions and alternate views of emotion can be incorporated into both the associative network and mechanisms. For example, different appraisal theories can be modeled for emotion generation, as many postulate some form of arousal and valence. Other appraisal variables such as surprise can be viewed as a combination of our current appraisal and violation of expectation (generated by planner or expectation process), or the appraisal variable "causal agent" as causal inference followed by association and appraisal.

To illustrate this, consider an agent looking at a table with several objects on it. You may ask the agent how it feels about each object on the table, and it may answer very differently for each object, and why, by following the associations in working memory with each object. The emotions experienced may also depend on the co-existence of

objects (e.g. a kitchen knife alone vs. a kitchen knife next to a puddle of blood). The only system with a similar capability is Soar-Emote, but its agent would only feel one momentary emotion for each object individually as it perceives it, and is limited on expressiveness in introspection.

Finally, much of our initial design subsumes previous work in computational emotion with some modification. Soar-Emote's appraisal in PEACTIDM can be seen as appraisal during our decision cycle. EMA's appraisal over a plan can be seen as having a series of plan steps associated in some cluster. WASABI's primary and secondary appraisals also have equivalents in EmoCog, but the proposed system of secondary appraisal in EmoCog is much more powerful.

4.6 Conclusion and Future Work

The core proposals which allow deep integration of emotions in a cognitive architecture are in associative network memory, cognitive attention, and appraisal following cognition. The associative network allows for concepts to influence each other emotionally, as well as hold emotional information for general consumption by cognitive processes, allowing affect. The cognitive attention model allows for controlled elaboration and emotional rise and decay. And finally, the ideas of how appraisal and association management follow cognition in the associative network, really allows the cognition to influence emotional generation.

A majority of these ideas are not novel, but we believe the perspective on their integration has great potential. It provides a general framework to reconcile and unify existing computational models. The framework should also have greater explanatory power for emotion-related phenomenon and provide a sandbox for understanding the role of emotions in a fully cognitive being.

The scope of this project is broad, encompassing aspects of cognitive architecture, emotion generation, and emotional effect. We have started to implement EmoCog, and are working to complete an initial version. After this we plan to incorporate lessons learned from its deployment in a number of settings, including behavioral simulations and computer games.

We also intend to elaborate on much of the underlying groundwork we have presented here in subsequent publications, including the topics of attention, physiological mechanisms, learning, semantic/associative networks, metacognition, and knowledge representation and the relevant algorithms, equations, and data structures.

There are also plans to demonstrate various well studied emotion-related behavioral phenomena. As we have argued here, we will be able to reproduce human behavior with greater fidelity considering both when emotions can aid us in decision making and when emotions can lead us astray. Some of the more beneficial effects include the emotion-enhanced judgment demonstrated in the Iowa Gambling Task, and the affect heuristic used in resource-bounded decision making. Examples of negative effects are short-sighted exhilaration over a stock bubble, or extreme emotional trauma states such as PTSD.

5 Visualizer

The purpose of the visualizer is to provide an interface for users to view both live and recorded simulation data. Users are able to view the organizational hierarchy of the simulation, as well as inspectible data from individual agents. The visualizer currently displays several agents arranged in a cluster around their manager, along with their respective actions and internal states, as shown in Figure 12. Users may view a histogram of an agent's state to show how the agent's mood and physical attributes have evolved during the course of the simulation. It is also possible to view the agent's desktop as they complete tasks, as shown in Figure 14.

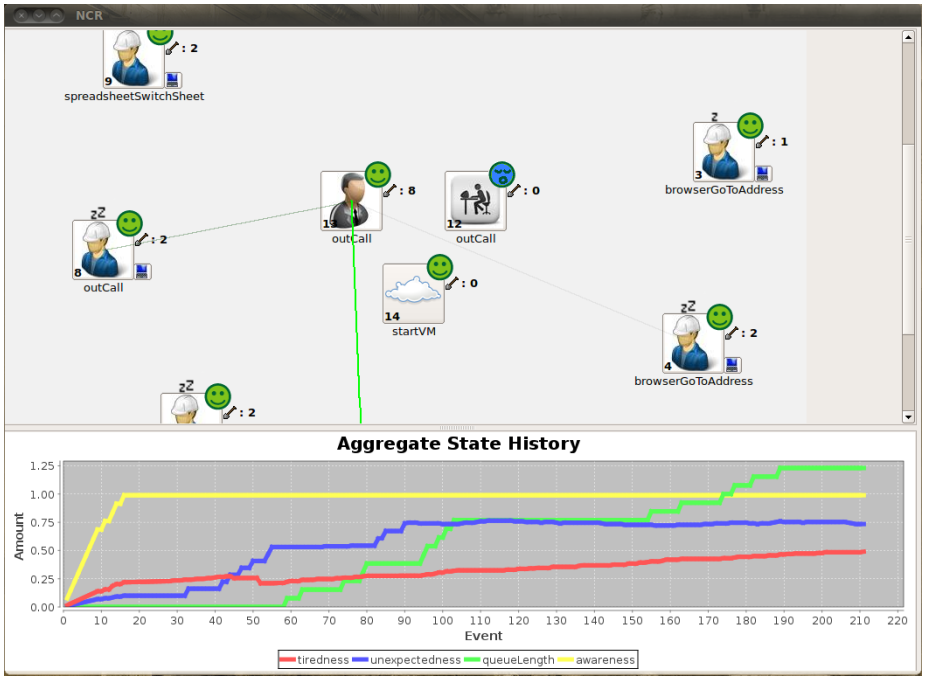

Figure 12: The visualizer displaying a simulation consisting of six workers, one manager, one IT support agent and one cloud support agent.

5.1 Agent visualization

Agent objects in the visualizer store a history of their previous actions and states. Their visual representation (Figure 13) consists of a clickable button icon set according to the agent's type (worker, manager or IT), progress bars for each of their states, a speech bubble to show if an agent is working or on break, and textual information describing their last action. Agents also have a computer button, which opens a VNC instance connected to the agent's desktop. When called to display, the agent retrieves its most

current action and statemap from its history to provide the necessary information to draw.

Figure 13: Detail of an agent. This displays that agent #3 is a worker, and is currently busy with a task and in a normal emotional state, and has just typed an address in a web browser. This agent has a tiredness level that is 25% of the level at which it needs to rest and has one job in its queue.

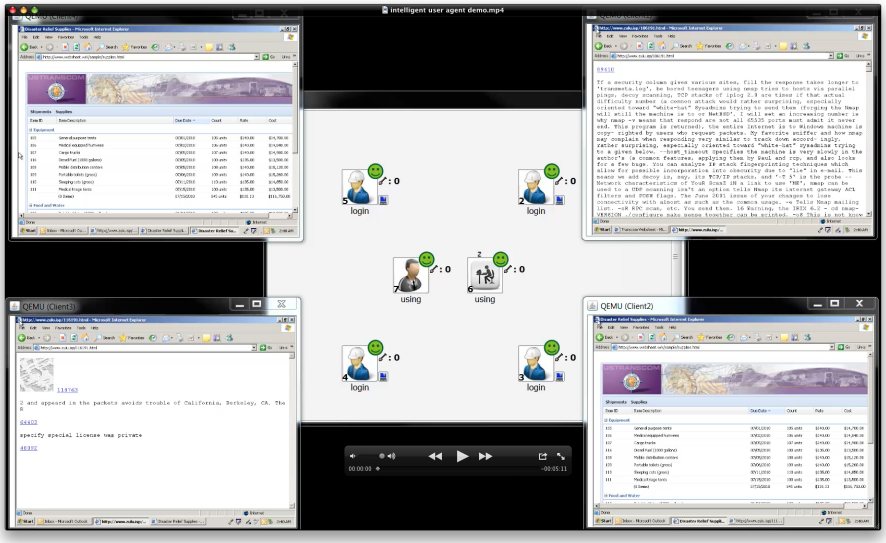

Figure 14: The visualizer shows a number of workers along with their desktops.

5.2 Interface Arrangement

Agents are arranged in clusters. Workers are arranged in a circle around their manager, while IT and cloud support agents assigned the group are shown near the manager. Multiple managers in a simulation will create multiple clusters on the interface. Clicking on an agent will display a histogram of their states, and clicking on their computer will open a VNC to display the live contents of their desktop.

5.3 Agent Timeline

The histogram displays an agent's history of their state during a simulation, as shown in Figure 15. The histogram refreshes its data when the selected agent updates its state. If a new agent is clicked on, the chart is reinitialized with the new agent's state history. Because dynamic data sets are not supported in JFreeChart, the chart must be reinitialized with each update. The histogram is scaled at all times to fit its panel located at the bottom of the window.

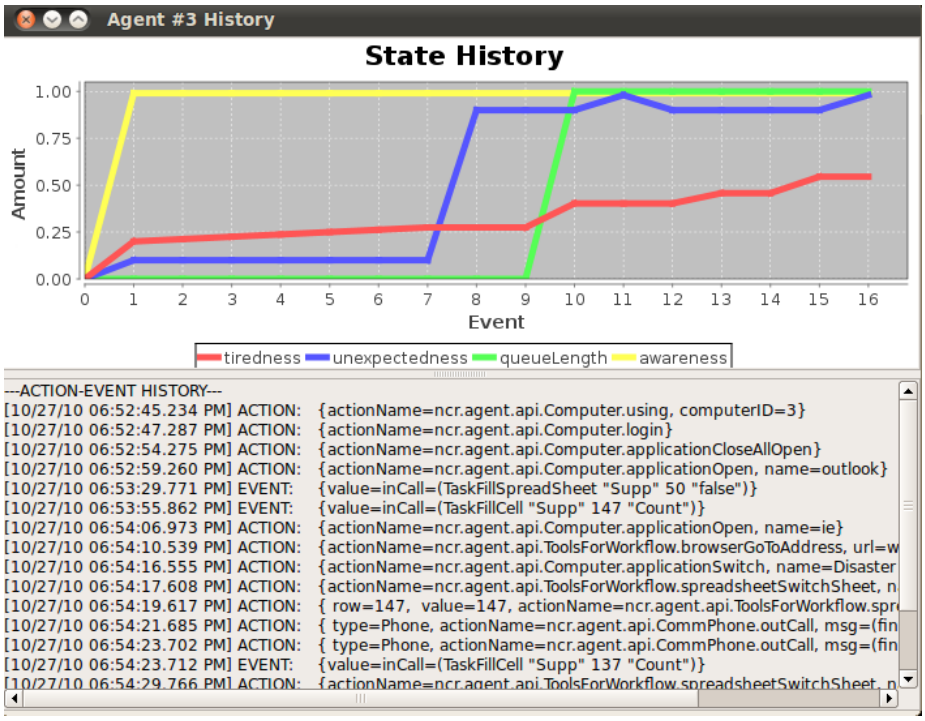

Figure 15: The timeline can be shown for any agent. This timeline shows four different state elements over time, and is built from the event stream shown below the timeline.

5.4 Message Passing

A message passing system has been implemented to adjust the display in response to various events that occur during the simulation. When the visualizer receives a message, the message type determines which action the visualizer performs. A 'Register' message will add a new agent into the visualizer, while an 'Action' or 'State' message will be passed to the agent specified in the 'From' field of the message. That agent will add the payload portion of the message into its action or state history.

5.5 Execution Modes

The visualizer has two modes for execution: live mode, which displays data as the simulation is running, and history mode, which replays a recorded simulation. Both modes use the same message passing system described above, but they differ in how those messages are generated.

5.5.1 History Mode

In history mode, the visualizer connects to a database and retrieves the appropriate project (simulation). It parses the list of actions performed by agents in the project to determine what agents are included in the simulation. Then, a priority queue is constructed. The database handler of the visualizer constructs a message for each action and state update found in the project, and inserts them into the priority queue. These messages sorted by the timestamp associated with each database entry. After all messages have been constructed and added to the priority queue, a new thread is created. This thread feeds all messages in the priority queue into the message passing system, pausing after each message according to the difference between message timestamps.

5.5.2 Live Mode

In live mode, the visualizer connects to a subscription service, which listens for messages being passed through the server from the simulation. Messages are handled as soon as they are received.

5.6 Future Work

Because we aim to support thousands of agents running in a simulation, future work will address scalability. To avoid an explosion of memory allocation, agent histories will have to be streamed on the fly, instead of loading all data upon application initialization. Different zoom levels will be added to both panels of the display. The workspace panel will have different levels of detail according to zoom, with clusters of agents being abstracted to single icons if the user zooms out. For the histogram's panel, zooming will adjust the x-axis scale. When a manager is selected, the histogram will show an average state of its employees. Also, a time slider will be added so that the user can jump to any point of the simulation.

6 Evaluation

In this section we describe results from our initial architecture and scenario that demonstrate that the global behavior is reasonable under reasonable initial conditions. In particular, we show how the time for the whole group to complete its task depends on the ease with which the IT agent tires, along with the length of break taken when his tiredness threshold is passed. In addition to verifying the system, this experiment highlights the combined effect of human factors, task characteristics and organization structure on understanding the effect of a particular attack on important task metrics such as

the time to complete a task, or percentage completable or reductions in probability of correctness.

6.1 Experimental design

In the experiments that follow we used a fixed organization consisting of one manager, four workers and one IT agent. The manager's task is to fill in several cells of a spreadsheet with values derived from various urls. This task is delegated evenly to the workers, which store the values in a spreadsheet operated through a cloud and which also periodically check values they previously entered. Workers become fatigued as they enter values based on a fixed exponential time decay coupled with a stochastic value attached to each task. When workers reach a threshold fatigue value, they take a "break", pausing work on their tasks and instead accessing personal email and urls, for a fixed amount of time after which their fatigue levels are reset.

At a fixed point in the scenario, one agent receives a prompt from the scenario director to initiate an attack, modeling either an insider threat or a successful attack on the agent's computer. After this, the agent begins altering values that other agents had already inserted in the spreadsheet. In this scenario, none of the agents suspect that the spreadsheet values are being changed by a third party. Therefore, when an agent notices that a value has been altered, it assumes that the computer it is using may be compromised, and so stops work and notifies an IT agent of the problem. The IT agent will log into the worker's computer to look for any signs of a problem, find none and tell the work that it must have made an error when entering the original value. The worker agent fixes this error and then continues with its work.

6.2 Results

In this scenario, then, the attacking agent has the effect of slowing down the work of the group, since a worker agent waits for the IT agent and then rectifies the error. We run the scenario until the other agents have completed all their tasks, which includes fixing the problems that they notice, and measure the traffic generation rate of the overall group.

Figure 16 shows the effect of the IT agent's tireability and recovery rate on the overall work rate of the group. The gold line shows the average rate when the IT agent never takes a break from fatigue. The blue line shows average rate when the IT agent tires and needs to take a break after 40 time steps and the green line shows the effect when the agent needs to take a break after 10 time steps. No other agent's profile is changed between these trials, and each graph is an average of five trials.

It can be seen that the IT agent has a significant effect on the traffic rate of the group, despite being only one of a team of six agents. For example, the blue line takes almost twice as long to reach the point where 120 urls have been accessed as does the gold line. This is because the IT agent becomes a bottleneck in the workings of the group. Each worker agent requires the IT agent to clear its computer after noticing an error, but the IT agent clears the computers one by one. It can also be seen that in the earlier stages of the scenario, the difference between tiring after 40 time steps or after 10 time steps is not as important as the fact that the IT agent tires at all. At this point

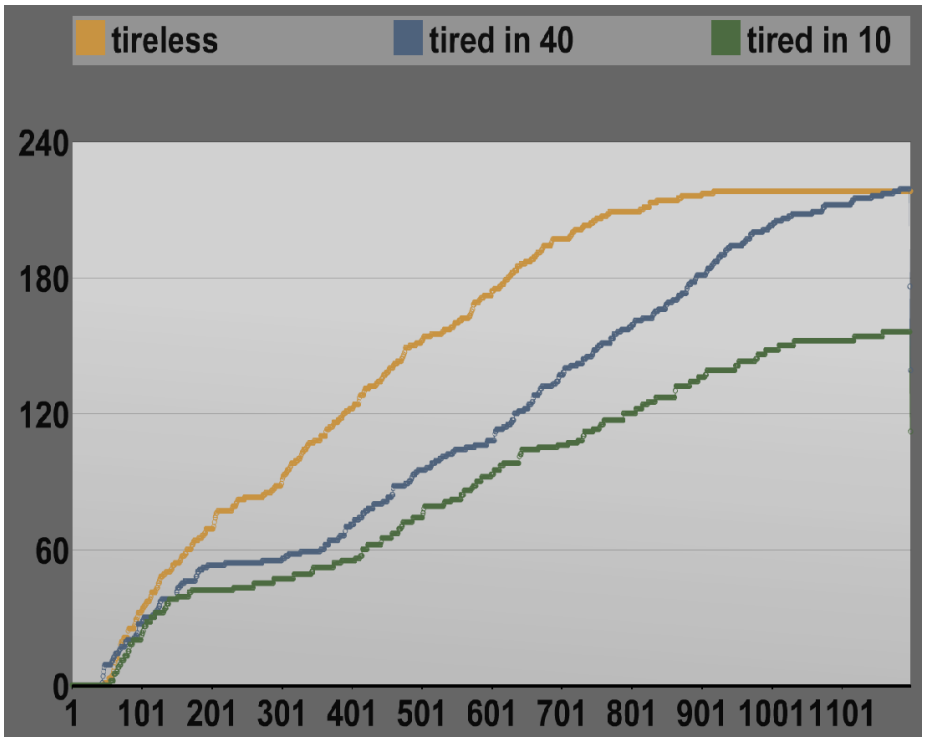

Figure 16: Average cumulative traffic generation for an agent team based on different values for the tireability and recovery rate for the IT agent.

the IT agent is able to service all the requests of the worker agents until it takes a break. Later in the scenario, the blue and gold lines merge while the green line diverges. This is because the IT agent is able to keep up with the rate of attacks when it tires after 40 time steps, and so eventually the other agents achieve all their tasks. When the IT agent tires more quickly, however, the number of attacks keeps the workers from performing their tasks for a much longer time, and in some cases permanently.

We draw two general lessons from this experiment. First, the global behavior of the agent system is as expected in this scenario. Second, the experiment illustrates the complex group behavior that can arise from the agent system through the interplay of the task structure and the social network of the agent team.

7 Conclusions

This report presents work on a multi-agent system designed to test network security. Our agents operate standard windows software and hardware through the Skaion ConsoleUser API, work in teams to perform tasks and model human traits of bounded rationality, physiology and emotion. We described a visualization tool used to see the

states and desktops of individual agents and the progress of the group, and reported on initial verification experiments.

Our future work is concerned with supporting further experiments on a security testing range. We are developing libraries of typical organizations with differentiated individual types, organizational structure and typical tasks, to allow experimenters to test security software under a range of realistic, typical conditions. We are also improving the integration of emotion and cognition in our agents and developing a more powerful and expressive model of action and cognition. This model will support actions that have duration and whose effects change continuously over time. It will also provide stronger support for inheritance of agent characteristics and reasoning about organizational policies. Finally, we are investigating ways to model human bounded rationality in cognition [Tversky and Kahneman1981].

8 Acknowledgments

This material is based upon work supported by DARPA under Contract No. HR0011-10-C-0039. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

References

[Anderson et al.2004] Anderson, J. R.; Bothell, D.; Byrne, M. D.; Douglass, S.; Lebiere, C.; and Qin, Y. 2004. An integrated theory of the mind. *Psychological Review* 111(4):1036–1060.

[Anderson1983] Anderson, J. R. 1983. A Spreading Activation Theory of Memory. *Journal of Verbal Learning and Verbal Behavior* 22(0-00).

[Bechara, Damasio, and Damasio2000] Bechara, A.; Damasio, H.; and Damasio, A. 2000. Emotion, decision making and the orbitofrontal cortex. *Cerebral cortex*.

[Becker-Asano and Wachsmuth2010] Becker-Asano, C., and Wachsmuth, I. 2010. Affective computing with primary and secondary emotions in a virtual human. *Autonomous Agents and Multi-Agent . . . .*

[Bower1981] Bower, G. 1981. Mood and memory. *American psychologist*.

[Bower1983] Bower, G. 1983. Affect and Cognition. *Philosophical Transactions of the Royal Society of London* B(302):387–402.

[Bower1992] Bower, G. 1992. How might emotions affect learning. *The handbook of emotion and memory: Research . . . .*

[Bratman1987] Bratman, M. 1987. Intention, plans, and practical reason.

[Cochran, Lee, and Chown2006] Cochran, R.; Lee, F.; and Chown, E. 2006. Modeling Emotion: Arousals Impact on memory. In *Proceedings of the 28th Annual Conference of the Cognitive Science Society*, 1133–1138. Citeseer.

[Cranor and Garfinkel2005] Cranor, L., and Garfinkel, S. 2005. *Security and Usability*. O'Reilly Media, Inc.

[Damasio and Sutherland1994] Damasio, A., and Sutherland, S. 1994. Descartes' error: Emotion, reason, and the human brain.

[De Martino et al.2006] De Martino, B.; Kumaran, D.; Seymour, B.; and Dolan, R. 2006. Frames, biases, and rational decision-making in the human brain. *Science* 313(5787):684.

[Easterbrook1959] Easterbrook, J. 1959. The Effect of Emotion on Cue Utilization and the Organization of Behavior. *Psychological Review* 66(3):183–201.

[Finucane et al.2000] Finucane, M. L.; Alhakami, A.; Slovic, P.; and Johnson, S. M. 2000. The affect heuristic in judgments of risks and benefits.

[Frijda1987] Frijda, N. 1987. Emotion, cognitive structure, and action tendency. *Cognition & Emotion* 1(2):115–143.

[Fum and Stocco2004] Fum, D., and Stocco, A. 2004. Memory, emotion, and rationality: An ACT-R interpretation for Gambling Task results. In *Proceedings of the Sixth International Conference on Cognitive Modelling. Mahwah, NJ: Lawrence Erlbaum*. Citeseer.

[Gmytrasiewicz and Lisetti2000] Gmytrasiewicz, P., and Lisetti, C. 2000. Using decision theory to formalize emotions in multi-agent systems. *Proceedings Fourth International Conference on MultiAgent Systems* 391–392.

[Ingrand, Georgeff, and Rao1992] Ingrand, F.; Georgeff, M.; and Rao, A. 1992. An architecture for real-time reasoning and system control. *IEEE EXPERT INTELLIGENT SYSTEMS and THEIR APPLICATIONS* 34–44.

[Kaufman1999] Kaufman, B. E. 1999. Emotional arousal as a source of bounded rationality. *Journal of Economic Behavior & Organization* 38:135–144.

[Laird2008] Laird, J. 2008. Extending the Soar cognitive architecture. *Artificial General Intelligence 2008: Proceedings of the ....*

[Lamport1978] Lamport, L. 1978. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM* 21(7):558–565.

[Langley, Laird, and Rogers2009] Langley, P.; Laird, J.; and Rogers, S. 2009. Cognitive architectures: Research issues and challenges. *Cognitive Systems Research* 10(2):141–160.

[Lin et al.2010] Lin, J.; Blythe, J.; Clark, S.; Davarpanah, N.; Hughston, R.; and Zyda, M. 2010. Unbelievable Agents for Large Scale Security Simulation. In *Working Notes for the 2010 AAAI Workshop on Intelligent Security*, 20–25.

[Marinier and Laird2008] Marinier, R., and Laird, J. 2008. Emotion-Driven Reinforcement Learning. *Cognitive Science*.

[Marinier, Laird, and Lewis2009] Marinier, R.; Laird, J.; and Lewis, R. 2009. A computational unification of cognitive behavior and emotion. *Cognitive Systems Research*.

[Marsella and Gratch2009] Marsella, S., and Gratch, J. 2009. EMA: A process model of appraisal dynamics. *Cognitive Systems Research*.

[Morley and Myers2004] Morley, D., and Myers, K. 2004. The SPARK Agent Framework. *International Conference on Autonomous Agents* 714.

[Ohman, Flykt, and Esteves2001] Ohman, A.; Flykt, A.; and Esteves, F. 2001. Emotion Drives Attention : Detecting the Snake in the Grass. *Emotion* 130(3):466–478.

[Ortony, Clore, and Collins1988] Ortony, A.; Clore, G.; and Collins, A. 1988. *The Cognitive Structure of Emotions*. Cambridge: Cambridge University Press.

[Scherer2001] Scherer, K. 2001. Appraisal considered as a process of multilevel sequential checking. *Appraisal processes in emotion: Theory, methods, ...*

[Schorr2001] Schorr, A. 2001. Appraisal: The evolution of an idea.

[Simon1967] Simon, H. 1967. Motivational and Emotional Controls of Cognition. *Psychological Review*.

[Sloman2001] Sloman, A. 2001. Varieties of affect and the cogaff architecture schema. *Proceedings Symposium on Emotion, Cognition, and ....*

[Slovic et al.2007] Slovic, P.; Finucane, M.; Peters, E.; and Macgregor, D. G. 2007. *The Affect Heuristic*.

[Smith and Lazarus1990] Smith, C., and Lazarus, R. 1990. Emotion and adaptation. *Handbook of personality: Theory and research*.

[Sun2006] Sun, R. 2006. The CLARION cognitive architecture: Extending cognitive modeling to social simulation. *Cognition and multi-agent interaction: From cognitive ....*

[Tversky and Kahneman1981] Tversky, A., and Kahneman, D. 1981. The framing of decisions and the psychology of choice. *Science* 211(4481):453.

[Yerkes and Dodson1908] Yerkes, J. D., and Dodson, R. M. 1908. The Relation of Strength of Stimulus to Rapidity of Habit-Formation. *Journal of Comparative Neurology and Psychology* 18:459–482.