Self-Preservation Mechanisms for Cognitive Software Agents

Uma Ramamurthy and Stan Franklin
Computer Science Division and Institute for Intelligent Systems
The University of Memphis, Memphis, TN 38152, U.S.A.
urmmrthy@memphis.edu and franklin@memphis.edu

***Abstract*-Humans and other animals have a sense of self-preservation that motivates them to take appropriate actions to preserve themselves. Intelligent software systems must have built-in or learnt functionality to detect emergencies in their domains, and must be able to act appropriately to save their state and data, and shutdown cleanly if necessary. They must also be able to restart and restore themselves to their most recent state once their computing environment comes back online. We have designed such mechanisms for an Intelligent Distribution Agent (IDA) built for the U.S. Navy. IDA implements Baars' Global Workspace Theory of consciousness. As a result, she can react to novel and problematic situations in a more flexible, more human-like way than traditional AI systems. We hypothesize that software agents like IDA must have self-preservation mechanisms to adapt and survive in their domains. This paper presents a design for self-preservation mechanisms consistent with IDA's cognitive cycle which can be implemented in intelligent software systems.**

I. Introduction

Software agents 'live' in computing systems and networks. They interact with their domains via messaging systems like email and chat-modes. When the computing environment in which they reside is about to crash, the agent must be able to detect such an emergency, save its data and state, and shutdown in a clean manner, if necessary. If the software agent runs out of memory, disk and/or processing resources, it must be able to negotiate with the system administrator to gain access to additional resources, or it must save its state and data before shutting down in a safe manner. Further, when the computing environment comes online again, the agent must read its saved state and data, and come 'alive' to function once again in a safe environment. All this requires a sense of *self-preservation* for software agents. Humans and other animals have such a sense of self-preservation. Building a sense of self-preservation into software agents will enable them to save their states, shutdown in emergencies and come 'alive' into their saved state when the computing system comes online once again. In this work, we detail designs for self-preservation in a software agent called IDA.

II. Global Workspace Theory and "Conscious" Software Agents

An autonomous agent is a system situated in, and part of an environment. It senses that environment, and acts on it, over time, in pursuit of its own agenda. It acts in such a way as to possibly influence what it senses in the future [8]. In that sense, the agent is structurally coupled to its environment [16], [17]. Biological examples of autonomous agents include humans and most other animals. Non-biological examples include some mobile robots, and various computational agents, including artificial life agents, software agents and computer viruses. This research is concerned with autonomous software agents 'living' in real world computing systems.

Autonomous software agents, when equipped with cognitive features chosen from among multiple senses, perception, working memory, transient episodic memory, declarative memory, attention, planning, reasoning, problem solving, learning, emotions, moods, attitudes, multiple drives, etc., are called cognitive agents [7]. Such agents promise to be more flexible, more adaptive, more human-like than typical currently existing software because of their ability to learn, and to deal with novel input and unexpected situations.

One way to design such cognitive agents is to model them after humans. We have designed and implemented such cognitive agents within the constraints of the Global Workspace Theory of consciousness, a psychological theory that gives a high-level, abstract account of human consciousness and broadly sketches its architecture [3], [4]. We call agents that are so designed "conscious" software agents.

Global Workspace Theory postulates that human cognition is implemented by a multitude of relatively small, special purpose processors, almost always unconscious. Coalitions of such processors find their way into a global workspace and hence into consciousness. This limited capacity workspace serves to broadcast the message of the coalition to all the unconscious processors, in order to recruit other processors to join in handling the current novel situation or problematic situation. All this takes place under

the auspices of contexts: goal contexts, perceptual contexts, conceptual contexts, and/or cultural contexts. Each context is itself, a coalition of processors. "Conscious" software agents should implement the major parts of Global Workspace Theory, and remain within its constraints. IDA is one such "conscious" software agent.

III. IDA AND HER ARCHITECTURE

IDA (Intelligent Distribution Agent) is a "conscious" software agent developed for the U.S. Navy. At the end of each sailor's tour of duty, he or she is assigned to a new billet. This assignment process is called distribution. The Navy employs some 300 people, called detailers, full time, to effect these new assignments. IDA's task is to fully automate this process, by playing the role of a human detailer.

IDA deals with both communication problems and constraint satisfaction problems. She communicates with sailors via email in natural language, understanding the content. She accesses a number of existing Navy databases, again understanding the content. She ensures that the Navy's needs are satisfied, for example, that there is the required number of sonar technicians on a destroyer with the required types of training. She holds down moving costs. And, she caters to the needs and desires of the sailor, as well as is possible. Keeping in focus the sailor's preferences and the Navy's needs, she looks up the Navy's database of available postings and generates a short list of jobs that are appropriate to offer to the sailor. In that process, she performs constraint satisfaction and deliberates over the various jobs that are available. Once the list of jobs is generated, she negotiates with the sailor to assist him in selecting his next job posting, and ensures that he has the required training he needs for that posting.

IDA is also intended to model a broad range of human cognitive functions. Her architecture is comprised of a number of different, but tightly integrated, modules each devoted to particular cognitive processes as shown in Fig. 1 below. Detailed descriptions of these modules comprising IDA's architecture are available in published papers [9], [10]. Here, we briefly describe some of the modules that directly relate to her self-preservation mechanisms and IDA's cognitive cycle, while providing references to more detailed descriptions.

A. IDA's Perception

Her perception module [24] is based on the Copycat architecture [11], [12]. IDA senses her world using three different sensory modalities: She receives email messages in natural language, she queries and reads database query-outputs and, lastly, she senses via operating system commands and messages.

In sufficiently narrow domains, natural language understanding is possible with an analysis of surface features without the use of a symbolic parser [13]. Allen describes this approach to natural language understanding as complex, template-based matching [1]. IDA's relatively limited domain requires her to deal only with a finite number of distinct message types, each with relatively predictable content. This allows for surface level natural language processing. Her perception module has been implemented as a Copycat-like architecture [12] with perceptual codelets that are triggered by surface features of her textual input. The module includes an input-workplace where the incoming stimulus is placed, a Slipnet that stores domain knowledge (providing perceptual contexts from Global Workspace Theory), a pool of perceptual codelets (processors from Global Workspace Theory) specialized for recognizing particular pieces of text, and production templates for building and verifying understanding. Together they constitute an integrated perceptual system for IDA, allowing her to recognize, categorize and understand. Using this perception module, IDA must also perceive the query-outputs read from databases as well as operating system messages. These latter two types of percepts of hers are much easier to process as they are better formatted than the natural language of the email messages.

B. IDA's Memory Systems

The IDA model has a number of different memory systems, including working memory, transient episodic memory and auto-biographical/declarative memory. Some of these memories are motivated by Sparse Distributed Memory system [14].

When IDA receives a message from a sailor saying that his projected rotation date is approaching and asking that a job be found for him, the perception module recognizes the sailor's name and social security number (SSN), and that the message is of the 'please-find-job' type. This information is then written to the workspace (working memory).

IDA employs a modified sparse distributed memory (SDM) [2], [21] for her major associative memories – transient episodic memory (TEM) and declarative memory (DM). SDM is a content addressable memory that, in many ways, is an ideal computational mechanism for use as a long-term associative memory. Any item written to the workspace cues a retrieval from both the TEM and DM, returning prior activity associated with the current entry. TEM and DM will be accessed as soon as the message information reaches the workspace, and the retrieved local-associations will be also written to the workspace.

At any given moment, IDA's workspace may contain, ready for use, a current entry from perception, prior entries in various states of decay, and local-associations instigated by the current or prior entries, i.e. activated elements of TEM and DM. IDA's workspace thus consists of both short-term working memory and something very similar to the long-term working memory of Ericsson and Kintsch [6].

C. IDA's "Consciousness" module

The apparatus for "consciousness" in IDA consists of a coalition manager, a spotlight controller, a broadcast manager, and a collection of attention codelets who recognize novel or problematic situations [5]. Attention codelets keep a watchful eye out for some particular situation to occur that requires "conscious" intervention.

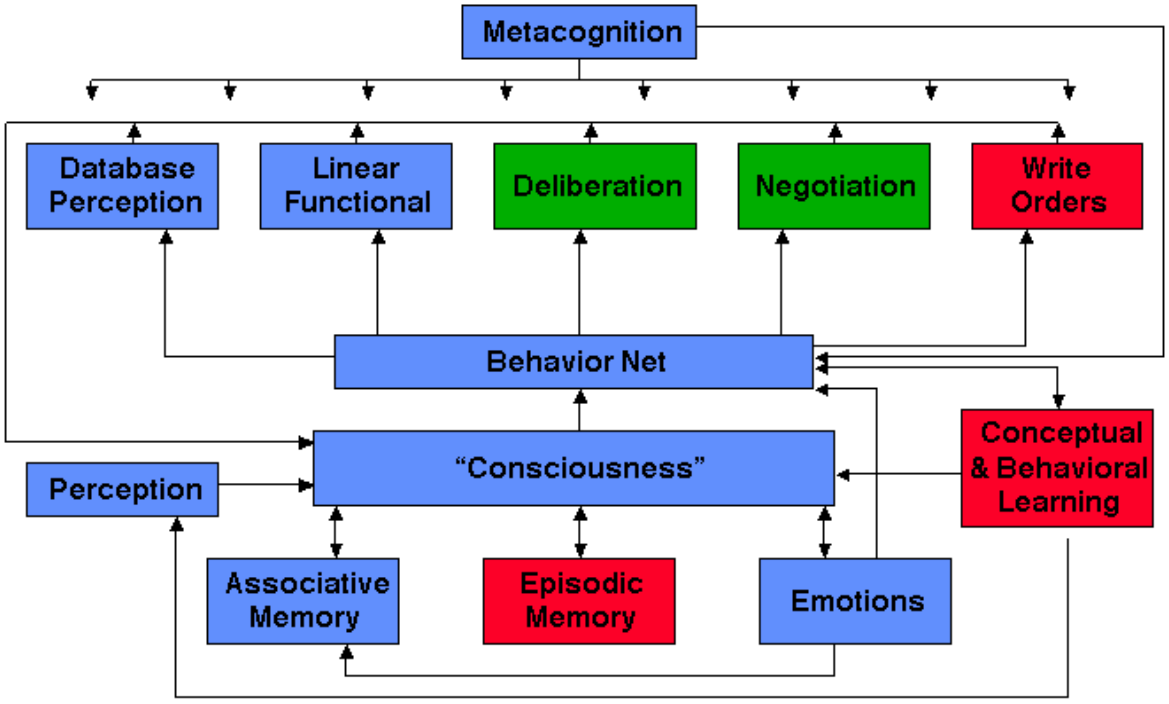

Fig. 1. IDA's Architecture

In most cases an attention codelet is watching the workspace, which will likely contain both external perceptual information and data created internally, including the local-associations. Upon encountering such a situation, the appropriate attention codelet will be associated with the small number of information codelets that carry the information describing the situation. This association should lead to the collection of this small number of codelets, together with the attention codelet that collected them, becoming a *coalition*. Codelets also have activations. The attention codelet increases its activation in order that the coalition, if one is formed, might compete for the spotlight of "consciousness". Upon winning the competition, the contents of the coalition are then broadcast to all codelets (Global Workspace Theory processors), the process of recruiting resources to solve the current problem.

For example, when an attention codelet sees the 'please-find-job' message type in the workspace, it gathers information codelets carrying the sailor's name, SSN, message type, etc., and forms into a coalition, which competes for "consciousness". If or when successful, its contents will be broadcast to all codelets.

D. IDA's Behavior Net

IDA selects her actions by means of an enhanced version of the behavior net [15]. This behavior net provides for high-level action selection in the service of built-in drives [19], that is, primitive motivators. She has several distinct drives operating in parallel. These drives vary in urgency as time passes and the environment changes. Behaviors are typically mid-level actions, many depending on several behavior codelets for their execution.

A behavior net is composed of behaviors, corresponding to goal contexts in Global Workspace Theory, and their various links. A behavior looks very much like a production rule, having preconditions as well as additions and deletions. It is typically at a high level of abstraction often requiring the efforts of several codelets to effect its action. Each behavior occupies a node in a digraph. The three types of links -- successor, predecessor and conflictor -- of the digraph are completely determined by the behaviors. Collections of related behaviors, together with their links, form behavior streams (goal context hierarchies in Global Workspace Theory). The behavior net is best thought of as a collection of instantiated behavior streams each connected to at least one drive.

As in connectionist models [18], this digraph spreads activation. The activation comes from that stored in the behaviors themselves, from the environment, from drives, and from internal states. The more relevant a behavior is to the current situation, the more activation it is going to receive from the environment. Each drive awards activation

to those behaviors that will satisfy it. Certain internal states of the agent can also send activation to the behavior net. One such example is activation from a behavior codelet responding to a "conscious" broadcast. Activation spreads from behavior to behavior along both excitatory and inhibitory links, and a behavior is chosen to execute based on activation and the satisfaction of its preconditions. Her behavior net produces flexible, tunable action selection for IDA. As is widely recognized in humans, the hierarchy of goal contexts is fueled at the top by drives, that is, by primitive motivators, and at the bottom by input from the environment, both external and internal.

When a "conscious" broadcast for a 'please-find-job' message is received by appropriate behavior-priming codelets, they instantiate a behavior stream in the behavior net for reading the sailor's personnel records. They also bind appropriate variables with sailor's name and SSN, and send activation to a behavior that knows how to initiate the access to the Navy personnel database. If or when that behavior is executed, behavior codelets associated with it begin to read data from the sailor's records. This data is written to the workspace. Each such write results in another round of local-associations from TEM and DM, the triggering of attention codelets, the resulting information coming to "consciousness," additional binding of variables and passing of activation, and the execution of the next behavior. As long as it is the most important activity going on, this process is continued until all the relevant personnel data are written to the workspace. In a similar fashion, repeated runs through "consciousness" and the behavior net result in a coarse selection of possibly suitable jobs being made from the job requisition database.

E. IDA's Cognitive Cycle

IDA functions by means of flexible, serial but cascading cycles of activity that we refer to as *cognitive cycles*. We will next explore the cognitive cycle in detail (as shown in Fig.2) in order to facilitate the reader's understanding of the material on self-preservation in Section V.

1. **Perception**. Sensory stimuli, external or internal, are received and interpreted by perception creating meaning. Note that this stage is unconscious.

    a. *Early perception*: Input arrives through senses. Specialized perception codelets descend on the input. Those that find features relevant to their specialty activate appropriate nodes in IDA's slipnet (a semantic net with activation).

    b. *Chunk perception*: Activation passes from node to node in the slipnet. The slipnet stabilizes bringing about the convergence of streams from different senses and chunking bits of meaning into larger chunks. These larger chunks, represented by meaning nodes in the slipnet, constitute the percept.

2. **Percept to Preconscious Buffer**. The percept, including some of the data plus the meaning, is stored in preconscious buffers of IDA's working memory.

3. **Local Associations**. Using the incoming percept and the residual contents of the preconscious buffers as cues, local associations are automatically retrieved from transient episodic memory and from declarative memory. The contents of the preconscious buffers along with the retrieved local associations from transient episodic memory and declarative memory together constitute long-term working memory [6].

4. **Competition for "consciousness"**. Attention codelets, whose job it is to bring relevant, urgent, or insistent events to "consciousness", view long-term working memory. Some of them gather information, form coalitions and actively compete for access to "consciousness". The competition may also include attention codelets from a recent previous cycle.

    The activation of unsuccessful attention codelets decays, making it more difficult for them to compete with newer arrivals. However, the contents of unsuccessful coalitions remain in the preconscious buffer and can serve to prime ambiguous future incoming percepts. The same is true of contents of long-term working memory that aren't picked up by any attention codelet.

5. **"Conscious" Broadcast**. A coalition of codelets, typically an attention codelet and its covey of related information codelets carrying content, gains access to the global workspace and has its contents broadcast. The current contents of "consciousness" are also stored in transient episodic memory. At recurring times not part of a cognitive cycle, the contents of transient episodic memory are consolidated into long-term associative memory.

6. **Recruitment of Resources**. Relevant behavior codelets respond to the "conscious" broadcast. These are typically codelets whose variables can be bound from information in the "conscious" broadcast. If the successful attention codelet was an expectation codelet calling attention to an unexpected result from a previous action, the responding codelets may be those that can help to rectify the unexpected situation. Thus "consciousness" solves the relevancy problem in recruiting resources.

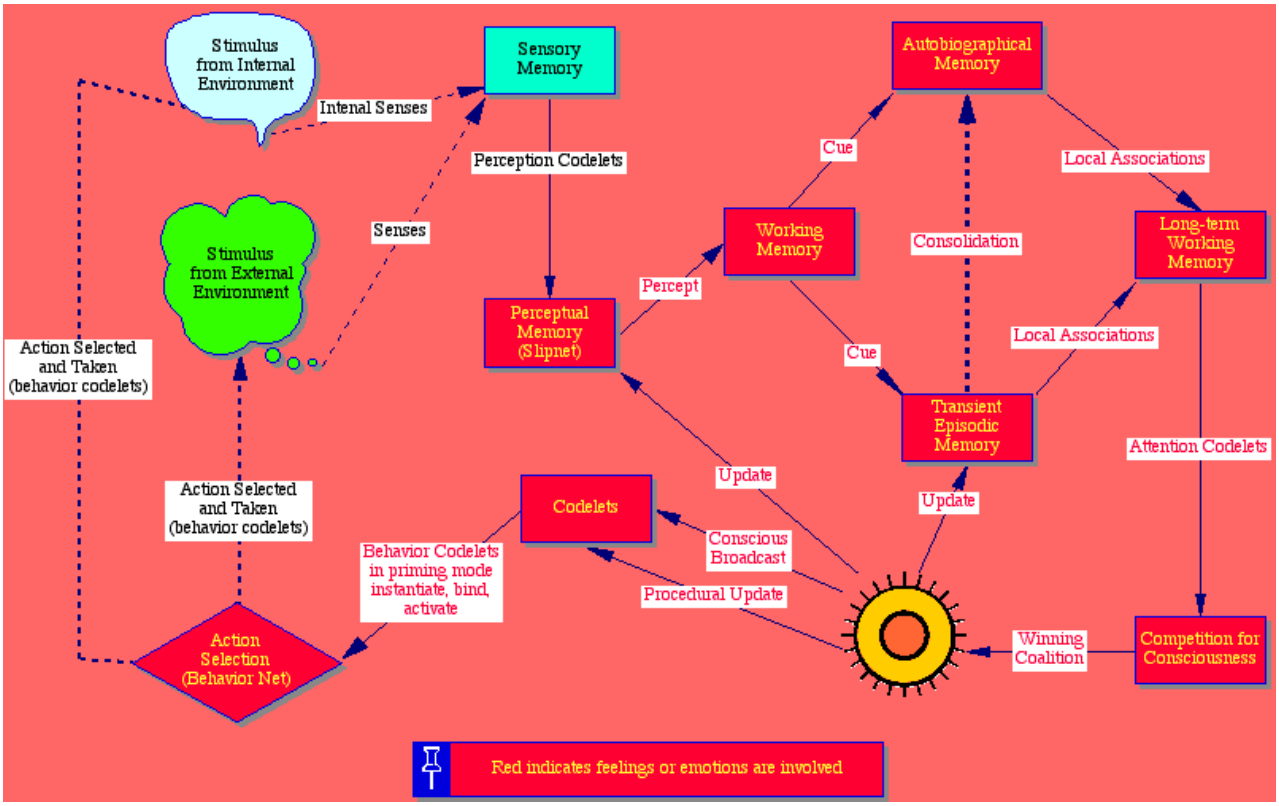

Fig. 2. IDA's Cognitive Cycle

7. **Setting Goal Context Hierarchy**. Some responding behavior codelets instantiate an appropriate behavior stream, if a suitable one is not already in place. They also bind variables, and send activation to behaviors. Here we assume that there is such a behavior codelet and behavior stream. If not, then non-routine problem solving using additional mechanisms is called for.

8. **Action Chosen**. The behavior net chooses a single behavior (goal context) and executes it. This choice may come from the just instantiated behavior stream or from a previously active stream. The choice is affected by internal motivation (activation from drives), by the current situation, external and internal conditions, by the relationship between the behaviors, and by the activation values of various behaviors.

9. **Action Taken**. The execution of a behavior (goal context) results in the behavior codelets performing their specialized tasks, which may have external or internal consequences. This is IDA taking an action. The acting codelets also include an expectation codelet (see Step 6) whose task it is to monitor the action, and to try and bring to "consciousness" any failure in the expected results.

IV. Need for Self-Preservation in Software Agents

The need for the kind of self-awareness required for self-preservation is beginning to be recognized by artificial intelligence researchers. DARPA (the Defense Advanced Research Projects Agency) is currently funding research on Cognitive Information Processing Technology (http://www.darpa.mil/baa/baa02-21.htm). The BAA (Broad Agency Announcement) says that "Cognitive systems have a number of differentiating characteristics, but in short, they can be characterized as systems that know what they are doing. … And, notably, they will be robust in the face of surprises provided by the environment; the avoidance of the "brittleness" of the expert systems of the past will be a distinguishing feature."

IBM motivates its notion of autonomic computing by noting that "a shortage of skilled administrators combined with the complexity of information processing technology infrastructure prevents us from taking full advantage of its potential." They offer as a solution "autonomic computing: a systemic view modeled after a self-regulating biological system." They go on to assert that "A virtual world is no less dangerous than the physical one, so an autonomic computing system must be an expert in self-protection. It must detect, identify and protect itself

against various types of attacks to maintain overall system security and integrity."
(http://www.research.ibm.com/autonomic/)

Aaron Sloman speaks of the need for "… a fast global alarm system to take control where rapid action is urgently required …" He builds such alarms into his CogAff meta-architecture for decision making agents [23]. The kind of self-preservation we are proposing here would seem to be a part of both cognitive computing and autonomic computing, as well as implementing Sloman's global alarms.

Computing systems in general, and software systems and software agents in particular must have system integrity and robustness. Systems that 'live' in computing systems and networks, and that interact with their environment via messaging systems, must have the capability to remember, recall and learn to adapt to their dynamic environments. This requires that the system have mechanisms to monitor itself and its environment, detect emergencies and act proactively to preserve its state and data. Such actions will enable the system to maintain its integrity and be robust by being able to shutdown in a safe manner during emergencies and restart itself with very little loss of data, if any.

Self-preservation mechanisms are essential for complex systems. Their complex domains, multiple modules and multitude of data structures make it a software engineering requirement that such complex systems have the ability to preserve themselves. Cognitive software agents, being complex systems, are specifically well-suited for having self-preservation mechanisms built into them.

V. Self-preservation mechanisms in IDA

IDA was designed and built as a proof of concept for Global Workspace Theory. She is designed to model many facets of human cognition. The mechanisms already built into her system make it possible to easily integrate the self-preservation mechanisms into her architecture. These mechanisms operate within IDA's continuing cognitive cycles. In this section, we describe the design details of IDA's self-preservation mechanisms. Each of these self-preservation mechanisms in IDA spans multiple and often overlapping cognitive cycles (detailed in Section III.E).

A. Saving data structures regularly

IDA's architecture has several data structures whose contents and current state are crucial for her functioning. The IDA model depends on its usual action selection mechanism involving its "consciousness" module and its behavior net to save these data structures on a regular, periodic basis to the hard-disk. Such saving ensures that the system's state is backed up to disk in case of any crash or emergency shutdown of the host system.

This backup mechanism begins with a *backup attention codelet* that gains activation as time passes from the last backup. This attention codelet competes to bring to "consciousness" the need to backup (step 4 of the cognitive cycle). When it eventually succeeds, perhaps in a subsequent cycle, a "conscious" broadcast occurs in step 5 of the cognitive cycle. The broadcast recruits behavior codelets in priming mode that instantiate a *backup behavior stream* whose individual behaviors are capable of the various parts of the backup (steps 6 and 7 of the cognitive cycle). This stream is attached to and receives activation from a *backup motivator drive*. The activation provided by the backup drive increases as time passes since the last backup. Each behavior in this stream competes for execution and, upon winning (step 8 of the cognitive cycle), executes its portion of the backup by releasing its *backup behavior codelets* in step 9 of the then current cognitive cycle.

The crucial data structures of IDA at any point in time are the input-workplace of her perception module, her workspace (working memory), her Transient Episodic Memory (TEM), her Declarative Memory (DM) and her behavior net. The backup behavior codelets save to the disk only the changes and additions since the previous save of the TEM and DM. This requires that the data items in TEM and DM have archive-bits in them. Every save of the contents of these two memories sets the archive-bits of their data items, thus enabling incremental-saving process for these two memory systems.

Though the backup process is initiated by a "conscious" process, it occurs so frequently that much of it becomes automatized over time, and occurs unconsciously [20].This backup is part of the self-monitoring process of the agent. The process ensures that IDA's latest state, or something close to it is always available on the disk, and in case of a power failure, the agent can come 'alive' again and restore herself to her last saved state. Thus, the agent's survival is ensured with minimal loss of information.

B. Negotiate for system resources

Amongst her self-preservation mechanisms is one whose job it is to monitor the host system that IDA 'lives' in. The codelets in this mechanism watch the host system's memory allocation and available memory; they watch the host system's disk space allocations and available disk space; and they watch the CPU load on the host system.

The starting point of this monitoring mechanism is a *monitoring attention codelet* whose activation increases with time past the last monitoring. In step 4 of the cognitive cycle, this monitoring attention codelet competes to bring to "consciousness" the need to monitor. When it eventually succeeds, the resulting "conscious" broadcast recruits behavior codelets in priming mode in steps 5 and 6 of the cognitive cycle. These codelets instantiate a *monitoring behavior stream* (step 7 of the cognitive cycle) whose individual behaviors are capable of the various functions of monitoring. This stream is attached to and receives activation from a monitoring drive. The activation provided by the monitoring drive increases as time passes since the last backup. Each behavior in this stream competes for execution and, upon winning (step 8 of the cognitive cycle), executes its portion of the monitoring by releasing its

monitoring behavior codelets in step 9 of the cognitive cycle.

These codelets execute the appropriate operating system commands to monitor and determine the status of the system resources. IDA's perception module has the built-in domain knowledge to understand the output of the operating system in response to those commands (step 1 of the cognitive cycle). In step 2 of the cognitive cycle, these understood percepts are written to the preconscious buffer of the agent's working memory. The working memory is watched by *monitoring expectation codelets* (specialized attention codelets) that were released by the *monitoring behaviors*.

When these monitoring expectation codelets sense through these system responses that the memory and disk-space availability is low and the agent may not be able to have the required resources to do her job, they compete to bring knowledge of this situation to "consciousness" (step 4 of the cognitive cycle). Under such conditions the coalitions formed by such monitoring expectation codelets would have high activation and would be expected to win the competition. When one wins, the resulting "conscious" broadcast recruits behavior codelets in priming mode (steps 5 and 6 of the cognitive cycle). These codelets instantiate a *negotiating behavior stream* whose individual behaviors are capable of carrying on an email correspondence with the system administrator via the agent's perception module. This stream is attached to and receives activation from IDA's *self-preservation drive* which always provides high activation (during step 8 of the cognitive cycle).

Through the behaviors in this negotiating behavior stream, IDA sends an email to the system administrator warning him/her about the low resource availability and, perhaps with subsequent emails, negotiates with him/her proactively to have the required resources made available (step 9 of the cognitive cycle). This process of negotiation with the system administrator may continue through several email messages, spanning multiple cognitive cycles.

IDA has built-in capabilities for communicating with sailors by email in natural language. These capabilities are extended to include negotiation with the system administrator in the interest of self-preservation.

C. Handling scheduled host system shutdown

When IDA receives scheduled system shutdown email messages from the system administrator, her perception module understands these messages in step 1 of the cognitive cycle. When such messages are perceived and written to IDA's workspace in step 2 of the cognitive cycle, a *self-preservation attention codelet* detects them, forms a coalition with information codelets, and competes for "consciousness" in step 4 of the cognitive cycle. Eventually the day and time when the system shutdown is scheduled come to "consciousness" by virtue of a "conscious" broadcast occurring in step 5 of the cognitive cycle. Behavior codelets in priming mode respond to the "conscious" broadcast and instantiate a behavior stream to deal with the scheduled shutdown (steps 6, 7 and 8 of the cognitive cycle).

In step 9 of the cognitive cycle, the chosen behavior is executed that spawns an *expectation codelet* which watches the system clock for the correct time to bring about a safe shutdown of the agent. Sufficiently ahead of that scheduled shutdown time, this expectation codelet gives itself high activation and forms a coalition to come to "consciousness" (steps 4 and 5 of the cognitive cycle). As a result of the "conscious" broadcast, the appropriate behavior stream is instantiated (steps 6 and 7 of the cognitive cycle) and the behaviors in that stream activate the appropriate *self-preservation codelets* to save all the important data structures of the agent to the disk (steps 8 and 9 of the cognitive cycle). One of the behaviors in that stream is the *agent-shutdown behavior*. Once the important data structures are saved to the disk, the agent-shutdown behavior activates to safely shutdown IDA ahead of the scheduled host system shutdown in step 9 of the cognitive cycle (surely in a later cycle than the one during which the scheduled shutdown day and time information came to "consciousness.").

When the host system comes back up, IDA is started automatically and she reinitializes her data structures from the data saved before the host system shutdown. This ensures the agent's survival with no loss of information.

D. Handling emergency host system shutdown

When the host system encounters an emergency, the operating system sends out a system message to all processes running on the host system about the oncoming immediate system shutdown. In this case, IDA's *self-preservation codelets* directly detect those messages after perception and go into action to preserve the agent's state and data. In this scenario, there is no "conscious" broadcast and codelets go into action in a reflexive mode. They activate the *backup codelets* in the various modules that are responsible for saving the important data structure in those modules. Once the important data structures are saved, the relevant self-preservation codelets directly activate behavior codelets responsible for shutting down the agent in a safe manner. These behavior codelets facilitate a clean shutdown of IDA, similar to proactive shutdown described in Section V.C.

Note that this process of dealing with emergencies in the host system requiring a shutdown occurs completely unconsciously. The cognitive cycle stops after step 2 to accomplish this process with codelets directly activating one another. To that extent, it is as if the behaviors involved were automatized [20] though no learning occurred. We've just described a particular case of a global alarm system a la Sloman [23]. When the host computer comes back online, again IDA starts up automatically and reinitializes her data structures with the saved data.

VI. Conclusion

The self-preservation system described herein for IDA adds significantly to the robustness of the system. It also takes her one additional step toward being what DARPA calls a cognitive system. The same techniques can be employed to add self-healing to IDA. Thus it is also a step in the direction of IBM's autonomic computing. And, a portion of IDA's self-preservation system implements Sloman's global alarm.

Being able to implement the additional capability of self-preservation mechanisms within the existing IDA architecture and its iterating cognitive cycle with only existing structures demonstrates, once again, the suitability of this architecture for developing highly complex and intelligent software agents. We have also discussed the need for such a self-preservation system for software agents, and described the various situations in which such systems would come into play.

VII. Future work

In the next stage of this research, we plan to implement this design into the computational IDA and experiment with it. We also plan to provide IDA with various other aspects of a self. An autobiographical self can be readily accomplished through the declarative memory implementation in IDA [21]. Other aspects of a self, including self-concept, a volitional self, and a narrative self, should all prove amenable to implementation within IDA's existing architecture including her continually iterated cognitive cycle.

Acknowledgment

The first author is supported in part by NIH Cancer Center Support CORE grant, P30 CA-21765 and by the American Lebanese Syrian Associated Charities (ALSAC). The authors acknowledge the support of the Conscious Software Research Group.

References

[1] Allen, J. J. (1995) *Natural Language Understanding*. Redwood City CA: Benjamin/Cummings.

[2] Anwar, A., and S. Franklin. (2003) Sparse Distributed Memory for "Conscious" Software Agents. Cognitive Systems Research 4:339-354.

[3] Baars, Bernard J. (1988) "A Cognitive Theory of Consciousness," Cambridge: Cambridge University Press.

[4] Baars, Bernard J. (1997) "In the Theater of Consciousness," Oxford: Oxford University Press.

[5] Bogner, M., U. Ramamurthy, and S. Franklin. (2000) "Consciousness" and Conceptual Learning in a Socially Situated Agent," In Human Cognition and Social Agent Technology, ed. K. Dautenhahn. Amsterdam: John Benjamins.

[6] Ericsson, K. A., and W. Kintsch. (1995) Long-term working memory. *Psychological Review* 102:21–245.

[7] Franklin, Stan (1997) "Autonomous Agents as Embodied AI," Cybernetics and Systems' Special issue on Epistemological Aspects of Embodied AI, 28:6 499-520.

[8] Franklin, S. and A. C. Graesser (1997) "Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents," In *Intelligent Agents III*. Berlin: Springer Verlag.

[9] Franklin, S. (2001) Conscious Software: A Computational View of Mind. In *Soft Computing Agents: New Trends for Designing Autonomous Systems*, ed. V. Loia, and S. Sessa. Berlin: Springer (Physica-Verlag).

[10] Franklin, S. (2001) Automating Human Information Agents. In *Practical Applications of Intelligent Agents*, ed. Z. Chen, and L. C. Jain. Berlin: Springer-Verlag.

[11] Hofstadter, D. R. (1995), *Fluid Concepts and Creative Analogies*, Basic Books.

[12] Hofstadter, D. R., and M. Mitchell. (1994) "The Copycat Project: A model of mental fluidity and analogy-making," In Advances in connectionist and neural computation theory, Vol. 2: logical connections, ed. K. J. Holyoak, and J. A. Barnden. Norwood N.J.: Ablex.

[13] Jurafsky, D., and J. H. Martin. (2000) *Speech and Language Processing*. Englewood Cliffs, NJ: Prentice-Hall.

[14] Kanerva, P. (1988) "Sparse Distributed Memory," Cambridge MA: The MIT Press.

[15] Maes, P. (1989) "How to do the right thing," Connection Science 1:291–323.

[16] Maturana, H. R. (1975) "The Organization of the Living: A Theory of the Living Organization," International Journal of Man-Machine Studies. 7:313-32.

[17] Maturana, H. R. and Varela, F. (1980**)** "Autopoiesis and Cognition: The Realization of the Living," Dordrecht, Netherlands: Reidel.

[18] McClelland, J. L., D. E. Rumelhart, et al. (1986) *Parallel Distributed Processing*, vol. 1. Cambridge: MIT Press.

[19] Negatu, A., and S. Franklin. (2002) An action selection mechanism for 'conscious' software agents. Cognitive Science Quarterly 2:363-386.

[20] Negatu, A., T. L. McCauley, and S. Franklin. (Submitted.) Automatization for Software Agents.

[21] Ramamurthy, U., S. D'Mello and S. Franklin. (2004) Modified Sparse Distributed Memory as Transient Episodic Memory for Cognitive Software Agents. In *Procdeedings of IEEE International Conference on Systems, Man and Cybernetics (SMC2004)*.

[22] Sloman, A. (1999) What Sort of Architecture is Required for a Human-like Agent? In *Foundations of Rational Agency*, ed. M. Wooldridge, and A. Rao. Dordrecht, Netherlands: Kluwer Academic Publishers.

[23] Sloman, A. (1998) Damasio, Descartes, Alarms and Meta-management. In Proceedings Symposiumon Cognitive Agents: Modeling Human Cognition. San Diego: IEEE.

[24] Zhang, Z., S. Franklin, B. Olde, Y. Wan, and A. Graesser. (1998) Natural Language Sensing for Autonomous Agents. In *Proceedings of IEEE International Joint Symposia on Intellgence Systems 98*.