Cognitive Science Quarterly (2000) 1, ..-..

An Action Selection Mechanism for "Conscious" Software Agents[1]

Aregahegn S. Negatu[2]
Institute of Intelligent Systems and
Department of Computer Science
The University of Memphis

Stan Franklin
Institute of Intelligent Systems and
Department of Computer Science
The University of Memphis

Abstract

In this paper we describe an action selection mechanism for a "conscious" software agent. We discuss briefly the main cognitive modules of our agent architecture. Our focus is on the operational/functional details of the "consciousness" module and the action selection mechanism and how these two work together. We describe how events come to "consciousness," how "conscious" events prime and bind to relevant actions/action-plans, and how the most relevant behavior/action is selected and executed. Our mechanisms provide the flexibility to interleave actions that operate under different tasks. We consider the descriptions in this paper a listing of the various hypotheses about human cognition resulting from our design of the agent described.

Keywords: Cognitive modules, Goal-context, Attention, Motivation, Drive, Action Selection, Consciousness

Introduction

In recent years, there has been a revival of the scientific study of "Consciousness" (Baars 2002). In this paper we describe an autonomous agent architecture (Franklin and Graesser. 1997) that implements global workspace theory, a psychological theory of consciousness (Baars 1988, 1997). In particular we will

[1] The second author was supported in part by ONR grant N00014-98-1-0332. The authors wish to acknowledge essential contributions from the Conscious Software Research Group: Ashraf Anwar, Ramesh Aitipamula , Arpad Kelemen, Ravikumar Kondadadi, Irina Makkaveeva , Lee McCauley, Uma Ramamurthy, Alexei Stoliartchouk, and Zhaohua Zhang.

[2] Address for correspondence: 3725 Norriswood 373, Dept. Mathematical Sciences, The University of Memphis, Memphis, TN. 38152, USA. e-mail: asnegatu@memphis.edu

Received 30.06.03

concentrate on the agent's action selection mechanism and its interrelation with the "consciousness" mechanism. The agent is provided with build in drives from which it derives goals and sub-goals, its agenda. Our agent "lives" in a complex environment where more than one behavior is possible at a given time. Thus there is competition between behaviors. The agent must select one as the most appropriate and act on it. This is the action selection problem, what to do next (Franklin 1995).

Computational models have long been major, and perhaps indispensable, tools in cognitive science. Many of these model some psychological theory of a particular aspect of cognition, attempting to account for experimental data. Others aspire to be a general computational model of cognition, such as the construction-integration model (Kintsch 1998), SOAR (Laird et al. 1987), and ACT-R (Anderson 1990). Most of these computational models are computer simulations of subjects in psychological laboratories, and are capable of performing tasks at a fine-grain level of detail. The simulated data ideally fit the human data like a glove. The theories on which the simulations are based are periodically revised so that new simulations conform more closely to the data. The computational models are judged on how closely they predict the data. A model may also be judged by the amount of change required in core, as opposed to peripheral, parameters that are needed to fit the data. Alternatively, the models are evaluated on a course-grain level, by observing whether a number of qualitative predictions (i.e., directional predications, such as condition A > B) fit the data. These data fitting approaches to testing theories have been hugely successful, and account for a large body of what is now known in cognitive science.

In this paper, we propose another class of computational models, which fall under the general heading of autonomous software agents (Franklin & Graesser 1997). These agents are designed to implement a theory of cognition and attempt to automate practical tasks typically performed by humans. We have been developing two such agents that implement global workspace theory (Baars 1988, 1997), one with a relatively simple clerical task (Zhang et al. 1998b) and the other with a rather complex personnel assignment task (Franklin et al. 1998). These models do not merely produce output that solves a specific engineering problem, as do typical software agents like web bots. They have mechanisms that simulate human cognition and their design decisions generate hopefully testable hypotheses (Franklin 1997), thus potentially providing research direction for cognitive scientists. We consider the descriptions in this paper a listing of the various hypotheses resulting from our design of the agent described. Hopefully, they will prove of use to cognitive scientists.

Conscious software agent

An autonomous agent is a system situated in, and part of, an environment that senses its environment and acts on it, over time, in pursuit of its own agenda. Its acts in such a way as to possibly influence what it senses at a later time (Franklin and Graesser 1997). A "conscious" software agent is an autonomous software agent that implements global workspace theory (Baars 1998, 1997).

According to *global workspace (GW) theory* (Baars 1988), one principal function of "Consciousness" is to recruit the relevant resources needed for dealing with novel or problematic situations. These resources may include both knowledge

and procedures. They are recruited internally, but partially driven by stimulus input. Global workspace theory postulates that human cognition is implemented by a multitude of relatively small, special purpose processes, almost always unconscious. Communication between them is rare and over a narrow bandwidth. Coalitions of such processes find their way into a global workspace (and 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 in solving the current problem. 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 processes. There's much more to the theory, including attention, learning, action selection, and problem solving. "Conscious" software agents should implement the major parts of the theory, and should always stay within its constraints.

In general, contexts are stable coalitions of processes. Though they consist of unconscious processors, contexts influence access to the global workspace, constraining conscious content. In explaining our agent's action selection mechanism, we particularly focus on goal contexts, hierarchical goal contexts and dominant goal contexts. Goal contexts represent desired future states of a system. They evoke and shape actions and sub-goals by constraining processes that can help state transitions from the current (problem) state to eventually reaching those future states. Goal contexts are triggered by conscious events. Their unconscious influence, in turn, can cause other conscious events. To achieve a goal state, sub-problems must often be handled thus achieving sub-goal states. Goal contexts allow for carrying out tasks that need multiple conscious experiences where each one may correspond to the satisfaction of a sub-goal state. Goal contexts generally form a hierarchical goal structure, where goal contexts are nested under other goal contexts. A goal context is called a *dominant goal context* if it dominates the global workspace by effectively influencing access to "Consciousness". The *dominant goal hierarchy* is a goal hierarchy, which contains the dominant goal context. Later we will show how goal contexts, goal hierarchy, and dominant goal contexts are related to the action selection mechanism (ASM).

IDA

Our "conscious" software agent, called IDA (Intelligent Distribution Agent), is being developed for the US Navy (Franklin et al. 1998; Franklin 2001). 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, to effect these new assignments. IDA's task is to automate the role of detailer. IDA is intended as a proof of concept project for "conscious" software.

Designing IDA presents both communication problems and constraint satisfaction problems. She must communicate with sailors via email and in natural language, understanding the content and producing life-like responses. Sometime she will initiate conversations. She must access a number of databases, again understanding the content. She must see that the Navy's needs are satisfied, for example, the required number of sonar technicians on a destroyer with the required types of training. She must understand and abide by the Navy's some ninety policies regarding distribution. She must hold down moving costs and training costs. And,

she must cater to the needs and desires of the sailor as well as is possible. . Finally, she has to write the orders and send them to the sailor.

"Conscious" software agent architecture

IDA is intended to model a broad range of human cognitive function. Her architecture, shown in figure 1, is comprised of a number of different modules each devoted to particular cognitive processes such as perception, learning, action selection, associative memory, "consciousness," emotions, deliberation and language generation. Her computational mechanisms include variants and/or extensions of Maes' behavior nets (1990), Hofstadter and Mitchell's Copycat architecture (1994), Jackson's pandemonium theory (1987), Kanerva's sparse distributed memory (1988), and Holland's classifier systems (Holland 1986). Table 1 succinctly lists several of the underlying assumptions or hypotheses that guided the design of IDA. We particularly focus on the action selection mechanism (ASM) of the architecture, on which we specify more design assumptions or hypotheses in table 2. Some of the assumptions or hypotheses are self-explanatory and are not directly addressed in this paper.

In the modeling of "Consciousness" as postulated by global workspace theory, processors are implemented by codelets, which are small pieces of code, each an independent thread. Codelets are specialized for some simple tasks, and mostly work in coalitions to underlie high level constructs like behaviors. They also often play the role of demons waiting for a particular condition to happen and acting on it as per their specialization. There are several kinds of codelets, as we will introduce them in the coming sections.

Three components implement "consciousness": the coalition manager, the spotlight controller, and the broadcast manager (Bogner et al. 1999). These are augmented by an assortment of attention codelets. An *attention codelet* waits for an occurrence of a specific situation that might need the involvement of "consciousness." When such a situation is encountered, the attention codelet increases its activation level and forms an association with other codelets who carry detailed information about the situation. The *coalition manager* is responsible for forming and tracking coalitions of codelets. Such coalitions, comprised of a limited number of active codelets, are initiated on the basis of mutual association strength. The *spotlight controller* picks the coalition that is the most relevant at the moment based on the highest average activation level of its codelets. The content of this coalition comes to "consciousness" to be broadcast by the *broadcast manager* to all codelets.

IDA senses her world by receiving email messages in natural language, and by reading records from several databases. Her primary perceptual mechanism is based loosely on the Copycat Architecture (Hofstadter1995; Hofstadter and Mitchell 1994; Zhang et al 1998). The mechanism includes a slipnet that stores domain knowledge, a pool of codelets (processors) specialized for recognizing particular pieces of text, and production templates for building and verifying understanding. A node is a piece of declarative knowledge and its activation level represents the degree of relevance it has in a situation. Each codelet represents a procedural knowledge. In copycat mechanism, macro-level behaviors emerge from the dynamics of activation

spreading among slipnet nodes and actions by codelets. DUAL (Kokinov 1994b) and AMBR (Kokinov 1994b), which is based on DUAL, have a cognitive architecture close to that of copycat. The major difference is that in DUAL/AMBR networks, the nodes, called DUAL agents, incorporate the underlying codelets.

Ida's relatively limited domain requires her to deal with only a few dozen or so distinct message types, each with relatively predictable content. This allows for surface level natural language processing. An underlying assumption motivates our design decisions about perception appear in Table 1; but we will not defend it in this paper.

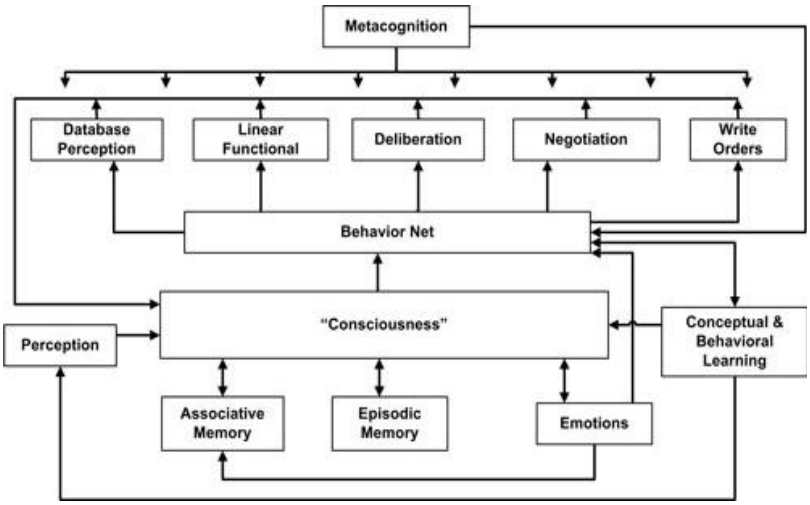

Figure 1: IDA's Architecture

IDA's perception process is augmented by sparse distributed memory (SDM) as her associative memory (Kanerva 1988). SDM is a content addressable memory that, in many ways, is an ideal computational mechanism for use as a long-term associative memory (LTM). Any item written to the workspace cues retrieval from LTM, returning prior activity associated with the current entry. At a given moment IDA's workspace may contain, ready for use, a current entry from perception or elsewhere, prior entries in various states of decay, and associations instigated by the current entry, i.e. activated elements of LTM. IDA's workspace thus consists of both short-term working memory (STM) and something very similar to the long-term working memory (LT-WM) of Ericsson and Kintsch (1995). As pointed out in the LTM hypothesis, what goes to LTM is a conscious content from working memory; *"Consciousness" is the gateway to transfer short-term memory content to LTM*.

IDA's design includes mechanism for emotion (McCauley and Franklin 1998; McCauley, Franklin, and Bogner. 2000). She may "experience" such emotions as guilt at not getting a sailor's orders out on time, and frustration at not understanding a message. Emotions influence all of her action selection (Damasio 1994) by altering the activations of drives, behaviors, and codelets. Emotions also influence

attention, the means of bringing certain content to "Consciousness". They also influence the strength of writes to LTM.

IDA deals with a number of hard and soft constraints that come from the Navy's needs and policies and the preferences of sailors. The mechanism involves a linear functional whose functions measure fitness with respect to an issue, and whose coefficients measure the importance of the issue (Franklin 2001).

Once possible jobs are identified for a given sailor, the question of getting the sailor out of his current position and to the new job within the prescribed time intervals remains. In between that are leave time, travel time, proceed time, and perhaps training. IDA deliberates by building possible temporal scenarios and choosing between them (Franklin 2000; Kondadadi and Franklin. 2001).

Table 1: Design assumptions & hypothesis in IDA architecture (not related to ASM).

Module/Component	**Design Assumptions or Hypothesis**
Perception	Much of human language understanding employs a combined bottom-up/top-down passing of activation through a hierarchical conceptual net, with the most abstract concept in the middle
Working Memory	The contents of perception are written to working memory before becoming conscious.
Long-Term Memory	What goes to long-term associative memory (LTM) is significant information from working memory. "Consciousness" is the gateway to transfer short-term memory content to LTM; a concept cannot be in LTM with out being conscious of it first.
Consciousness	Human "Consciousness" must have a mechanism for gathering processors (neuronal groups) into coalitions, another for conducting the competition, and yet another for global broadcasting.
Emotions	Action selection will be influenced by emotions via their effect on drives. Emotions also influence attention and the strength with which items are stored in the long-term memory.
Language Production	Much of human language production results from filling in blanks in scripts, and concatenating the results.

To resolve different issues, IDA negotiates with each sailor. Negotiation continues until IDA makes an assignment that is acceptable for both the Navy and the sailor. Once an assignment is final, IDA writes orders and the assignment becomes official. In the architecture, the "consciousness" and the behavior net modules play the central role. The tasks like negotiation and deliberation happen in the services of the action selection mechanism (ASM). We will discuss about the ASM and its integration with the "consciousness" system in detail in the coming sections.

In IDA, everything is done by codelets (simple processes); as a coalition they underlie many of the high-level computational constructs; thus IDA's architecture is conceptually in agreement with the Minsky's Society of Mind (1986). Sloman have proposed a human-like cognitive architecture, H-CogAff (1999), which has three layers: reactive, deliberative, and meta-management. Although, not explicitly shown in Figure 1, IDA's architecture has the necessary mechanisms for reactive responses. IDA's deliberation and negotiation cognitive functions correspond to the deliberative layer in H-CogAff. IDA's metacognitive module corresponds to the

meta-management layer. As in H-CogAff, each layer in IDA has a perceptual (internal/external), processing, and action (internal/external). So, IDA fits well in the schema of H-CogAff.

Action selection mechanism

Mind is best viewed as a control structure for an autonomous agent (Franklin, 1995). Its continual task is to produce the next action. Thus every agent must repeatedly solve the action selection problem. In the design and implementation of software agents, modeling, design and implementation of an action selection mechanism (ASM) plays a central role. Primary motivation is essential. An ASM chooses the next action so that the agent can satisfy one or more of its drives.

An ASM selects an action from many alternatives when it handles the many sub-problems it may encounter at one time. To choose the most sensible or appropriate action, an ASM must consider such factors as the urgency of the sub-problems, the importance of the related drive, the availability/unavailability of opportunities in the environment, and the desirability of pursuing a stream of action that has already been started. Tyrell (1993) accounted the requirements of action selection mechanisms in detail and we will not discuss about them here.

Drives

Drives are built-in or evolved-in primary motivators. All actions are chosen in order to satisfy one or more drives. A drive may be satisfied by different goals. While drives are invariant except for changes in intensity over time, the goals that satisfy them vary along with the internal state of the agent and with the environment.

A drive in IDA has an importance parameter that denotes its relative significance compared to other drives. The importance is a real value in (0,1], A drive with an importance value of 1 spreads the highest level of motivational activation to its underlying goals. If a drive has an instantiated goal structure hooked to it, the activation the drive spreads to the goal structure is weighted by the importance value.

Streams: action plans

Our ASM is based on Maes' work (1990) with important additions to it and to our earlier work (Song and Franklin. 2000). Maes' action selection mechanism (MASM), also referred to as a behavior net, has received considerable attention. Tyrell (1993) has investigated MASM and reported on its strengths and limitations. Dorer (1999), and Decugis & Ferber (1998) introduced a variation of MASM by modifying the domain representation or the network structure.

Neither MASM nor any of the above variations incorporate variables or variable passing in the behavior network; i.e., there must be a distinct behavior defined for each simple action. As a consequence, for complex domains the behavior network must have a prohibitively large number of behaviors. Maes (1990) argues that the introduction of variables will destroy the merits in her algorithm. Still, to apply the behavior net to complex domains, the introduction of variables is a necessity.

Rhodes (1995) proposed a variation of MASM that introduces variables using the indexical-functional aspects (Agre, P. & Chapman, D. 1987). He defines real

world functions in the task domain that help the agent to relate objects in its environment. This approach reduces design time considerations if the domain task can be captured in the behavior network with a relatively small number of functions compared to the number of objects in the environment. In the worst case, the number of functions could approach the number of items.

The introduction of classical variables to behavior network makes the selection control mechanism a hybrid (symbolic/connectionist) system. Song and Franklin (2000) presented this approach, and our work is based on this variation of MASM with some additional extensions. The variation of MASM in IDA is different in the following important ways. (1) It provides a more sophisticated hierarchical goal context system for our "conscious" agent. (2) It is implemented so that it could serve as a development tool for a general-purpose cognitive agent; i.e., easy to deploy the system for a new domain. (3) Streams, as partial action plans, can be organized hierarchically which will allow it to handle important classes of cognitive functions such as non-routine problem solving, action automation and learning of new behavior streams (Negatu & Franklin 1999).

Our ASM is a network of behaviors that are partitioned into connected components called streams. Each stream has one or more behaviors with stated goal(s), that is, dedicated to dealing with a particular sub-problem. A stream is a computational structure in the behavior net that can produce relevant stream of actions to satisfy a goal. One can also look at a stream as a partial action-plan that controls an agent's actions (internal or external). A stream is constructed as a directed graph, which has goal(s) and behavior(s) as nodes and various links connecting the nodes along which activation spreads as prescribed for a Maes' behavior network. Activation originates from drives, from the situation in the external world, and from internal states. A stream can have several goals and behaviors (figure 2).

Behavior

A behavior is comprised of a precondition-list, variable slots, activation, an action, an add-list and a delete-list. The precondition-list is a set of propositions that determines the relevance of the behavior to the current environmental and internal state. The variable slots are placeholders for specific concepts and/or objects that need to be bound for behavioral actions to take place. When all its propositions are satisfied, the behavior is *executable*. The activation is accumulated by the behavior in the dynamic process of activation spreading within the behavior net. When a behavior is executable and its activation level is above a threshold value, and then it competes globally for execution with other similarly qualifying behaviors. The winner, having the highest activation, determines the agent's next action. A behavior resets its activation-level immediately after its execution. The ASM has a number of global parameters that affect the dynamics and they are tuned so that the agent system can produce the expected domain functionality.

Behavior codelet: Codelets that underlie instantiated behaviors as a high-level constructs are called *behavior codelets*. They execute behavioral actions using the information in the bound variables. The action of a behavior produces changes in the internal state and/or in the environment. Each behavior also has an expectation of

what should happen when it executes. Its add-list and delete-list specify propositions that are expected to become true or false.

Goal

A goal node in a stream has all the components of a behavior except an action and variable slots; in addition, goals have satisfaction-condition component. A goal may need to be satisfied only once, or continuously, or continuously until a flag is raised. The satisfaction-condition is used to specify a situation that drops the motivational strength of a goal. A goal is satisfied when all the propositions in its precondition-list are true. A satisfied goal infers that all the propositions in its add-list are satisfied and those in its delete-list are unsatisfied. Goals do not act, they simply state when a particular state of affairs is desired and/or achieved.

Without the dynamics of activation spreading, behaviors and goals are simply rules. A behavior is a rule operator that tries to act when its precondition is satisfied, and its action produces an expected effect specified by its post conditions. Similarly, a goal is an inference rule that declares the achievement of its post conditions when its preconditions are satisfied.

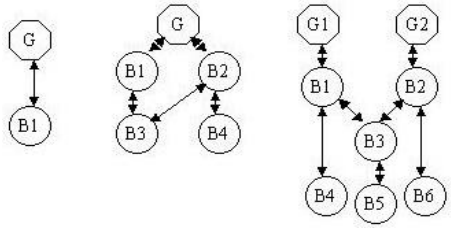

Figure 2: Examples of possible stream configurations.

Behavior network module: A hierarchical goal context system

According to global workspace theory goal contexts perform four major functions. (1) They help to evoke, shape and produce actions. (2) They allow the agent to deal with tasks that extend over more than one conscious event. (3) They represent future states of the agent serving to constrain the processes that can help reach those future states. (4) They have hierarchical structures that can constrain a stream of "Consciousness". In this section we discuss the high-level implementation of IDA's behavior net module that comprises her goal context hierarchy.

Goal contexts and hierarchical goal contexts

The behavior network, which has streams as its components, constitutes the goal context system of the "conscious" agent. A behavior in a stream represents a goal context. When this stream gets instantiated its behaviors eventually get executed one by one. Each executing behavior becomes the *dominant goal context*.

A stream gets its activation from one or more drives and/or other streams, as well as from the internal and/or external environment. That is, a stream gets activation from the drives that can be satisfied by its goals. A stream also becomes a source of activation to other streams whose goals can satisfy one or more of the preconditions

of its behavior(s). This means, there is a goal hierarchy where streams are nested under other streams.

Figure 3 illustrates an example of such a hierarchical goal context. Drives are at the top of the hierarchy where streams serve one or more of these drives directly or indirectly. Stream 1 satisfies the drive directly while streams 2 and 3 satisfy it indirectly through stream 1. That is, stream 1 receives activation from drive D while streams 2 and 3 get their top down activation from the drive through stream 1.

Behavior B3 has proposition(s) or sub-problems in its precondition list that can be satisfied or solved by stream 2. So, B2 of stream 1 spreads activation to the goal node of stream 2. In the same way, behavior B5 of stream 1 relates to stream 3.

Such hierarchical goal structures comprise the hierarchical goal contexts of the "conscious" system. Among many competing goal hierarchies, the one that has an executing behavior (dominant goal context) is the *dominant goal context hierarchy*.

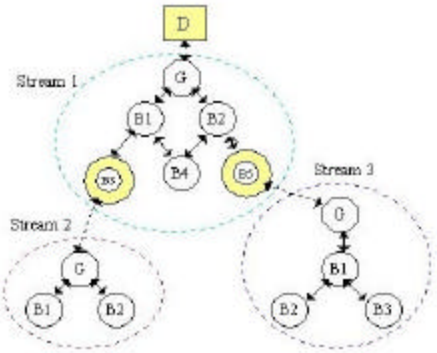

Figure 3: An example of a hierarchical goal context that has multiple streams working together. Stream 1 handles its problem only after streams 2 and 3 solve its sub-problems.

Behavior priming codelets

Behavior priming codelets listen to "conscious" broadcasts. They recognize information relevant to them, and know what streams to instantiate in the behavior network. When a behavior priming codelet finds itself relevant to a particular broadcast, it spawns a copy of itself with its variables bound to the information from the broadcast. Then the bound codelet jumps to the sideline (see the next section). Once there, it has three functions. (1) If necessary, it primes streams when the information it carries is relevant. (2) It binds the variables of related behaviors in the streams it primed. (3) It spreads environmental activation to all the behaviors for which it bound their variables. After a stream is primed, it will be instantiated only if another instance of that same stream in the same task context is not already instantiated.

Stream instantiation

A *behavior net template* contains instantiable behaviors, goals and streams with none of their variables bound. A *behavior/goal/stream template* is a behavior/goal/stream in the behavior net template. For a given sub-problem, one or

more streams can be relevant. When a stream is relevant, the instantiation mechanism (discussed below) creates an instance of the stream from its template that includes the binding of variables in its behavior(s) and goal(s) wherever possible.

Disposing of action plans

If an instantiated stream satisfies a drive/sub-goal, then this stream terminates itself as soon as it completes its actions. The associated codelets die soon afterwards. In a dynamic domain, it is possible that an instantiated stream may never get a chance to finish its actions. There needs to be a way to forget about such streams after some time. In IDA, streams (with their goals and behaviors), behavior codelets and behavior priming codelets are designed with a decay mechanism on their activations. As time passes, without they receiving additional activation, their activation decays; and all die when activation-levels fall below a threshold.

Working with "consciousness"

If the behavior network constitutes the active goal contexts of a "conscious" system, then it should be integrated to the "consciousness" module in order to effectively influence "conscious" activities and to play its action selection role.

Typically, an action performed by a dominant goal context brings about a conscious experience, and this conscious experience in turn could evoke a goal context whose action will influence the next conscious experience. To adhere to the global workspace theory, the behavior network system and the "consciousness" mechanism should be coupled in such a way that a conscious event (via its broadcast of information content) influences which behavior (goal context) becomes active (dominant) in the immediate future. In turn the execution of this dominant goal context should constrain what comes to "consciousness" next. Figure 4 will help us to explain how the action selection and "consciousness" modules are coupled and work together.

As per global workspace theory, a goal context is a coalition of processors. In this case the behavior codelets are identified with the behavior they underlie, but at a different level of abstraction. Behaviors and behavior codelets bring different levels of dynamics to the action selection process. Here's how it works.

Skybox

The skybox is an abstract place where instantiated streams reside. Instantiated streams form the active behavior network. Its dynamics is implemented by mechanisms in the skybox.

One or more streams can be instantiated at a time. If these streams are in the service of the same task, then the streams can be merged together as a single active behavior stream. In this case activation passes from one instantiated stream to another one via common behaviors. If any two streams are instantiated to service different tasks, then the streams are in the skybox at different levels. Streams at different levels do not share the activation dynamics. However, their behaviors compete globally in order to become the next dominant goal context. *BNManager* is the component module that implements the functions of the skybox.

Sideline

The sideline is an abstract level where instantiated behavior codelets reside. A behavior codelet gets its binding information and activation-level setting from its behavior. That is, a behavior in the skybox binds the relevant information to variables in its underlying behavior codelets in the sideline. In addition, the sideline has other functions. (1) Behavior priming codelets perform their tasks of priming, binding and sending activation to their relevant behaviors. (2) Heuristic rules are kept that enhance the accuracy of the priming decisions; (3) Bound behavior codelets form coalitions and jump to the playing field when their corresponding behaviors in the skybox become active. (4) Behavior codelets return to the sideline when they finish their tasks in the playing field. The *Sideline Manager* component of our action selection mechanism implements the sideline functions.

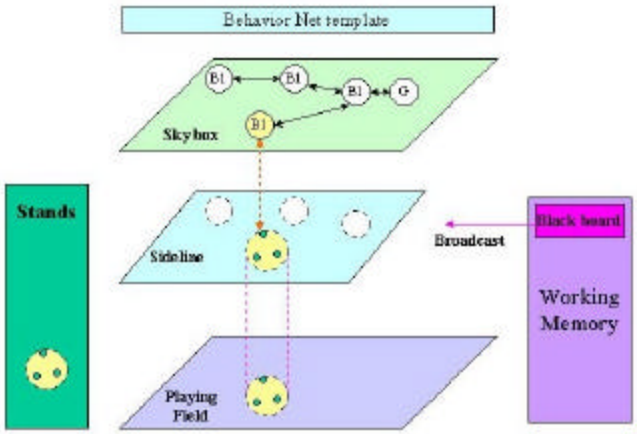

Figure 4: A behavior in action with its involved components of "consciousness" and the ASM module. Behavior-priming codelets in the stands receive a global broadcast of conscious content (from the blackboard) and prime a relevant stream; the stream gets instantiated in the skybox from its template. When the dynamics in the skybox selects a behavior, then its corresponding coalition of codelets act after jumping from the sideline to the playing field.

How all work together

Figures 4 and 5 illustrate how the behavior net module and "consciousness" work together. IDA starts, with all her mechanisms and built in knowledge distributed over its modules. For instance, the action selection mechanism has built in domain knowledge in the behavior net specification and in all the implemented behavior and behavior-priming codelets.

Attention codelets collect *information codelets* containing information about internal or external states of the agent and compete for "consciousness." The "consciousness" mechanism picks the winning coalition of attention and information

codelets, and broadcasts their content in order to recruit unconscious contexts that help to interpret the later conscious events.

Suppose, for example, IDA receives a message from a sailor saying that his projected rotation date (PRD) is approaching and asking that a job be found for him. The perception module would recognize the sailor's name and social security number (ssn), and that the message is of please-find-job type. This information would then be written to the working memory. The general principle here is that the contents of perception are written to working memory before becoming conscious.

An attention codelet will note the please-find-job message type, gathering information codelets carrying name, ssn, and message type, be formed into a coalition, and compete for "Consciousness". If or when successful, its content will be broadcast. Among behavior-priming codelets waiting in the stands, some find themselves relevant to the broadcast, make a copy of themselves with variables bound with information from the broadcast and jump to the sideline. Once there, they collectively or individually prime all the relevant streams. The *BNManager* retrieves all the primed streams from the template memory and instantiates them. Instantiated streams become part of the dynamics of the behavior net. Then the behavior-priming codelets bind behavioral variables, assert relevant propositions, and send environmental activation to the appropriate behaviors of the instantiated streams. At the same time, corresponding to each instantiated behavior, a coalition of behavior codelets, with the appropriate information bound, is instantiated in the sideline and waits for execution.

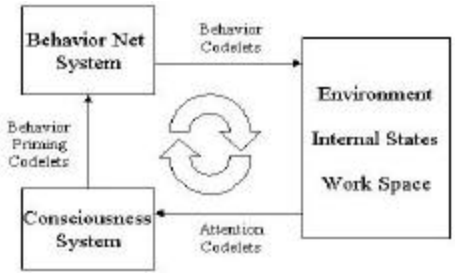

Fig. 5: Interleaving of unconscious goal contexts and their actions with "conscious" events. Attention codelets brings information to "Consciousness", "Consciousness" broadcasts conscious content, behavior-priming codelets receive broadcast and prime relevant behavior streams, and instantiated streams act to affect changes.

The dynamics of the behavior net eventually picks the dominant goal context (behavior), and its coalition of processes (the behavior codelets) waiting in the sideline jumps to the playing field and performs the designed action using the bound information. After finishing their execution behavior codelets return to the sideline and wait there as long as their corresponding behavior resides in the skybox.

Returning to our example, the broadcast is received by appropriate behavior-priming codelets, which will instantiate a behavior stream for reading the sailor's personal record. They also bind variables with name and ssn, assert propositions that can satisfy the precondition of the behavior and spread environmental activation. The behavior that knows how to access the database can be chosen by the dynamics

in the behavior net and execute. When the behavior executes, its behavior codelets read data from the sailor's file and write the data to the workspace. Each such writes results in another round of associations, the triggering of attention codelets, the resulting information coming to "consciousness," and the same way another behavior-priming codelets could influence the execution of the next behavior. As long as it's the most important activity going, this process continues until all the relevant personal data is written to the workspace. Such repeated runs through "consciousness" and behavior net result in course selection of possible suitable jobs being made from the job requisition database. The dynamics in the behavior net may be influenced by emotional "experiences" such as guilt at not getting a sailor's orders out on time, and frustration at not understanding a message.

IDA has a behavior stream for a constraint satisfaction system designed around a linear functional. It provides a numerical measure of the suitability of a specific job for a given sailor. Here issues like moving costs, Navy policy, etc. are considered. With the run of such stream, the job's fitness value is written to the workspace.

Table 2: Design assumptions & hypothesis in IDA architecture (not related to ASM).

ASM Component	**Design Assumptions or Hypotheses in the ASM System**
Motivation	The hierarchy of goal contexts is fueled at the top by drives, the prime motivators, at the bottom by input from the environment, both internal and external.
Significance of Action	An action attracts involvement of "Consciousness" in proportion to the motivation level of the dominant goal context that perform the action.
Voluntary Action	A timekeeper who becomes less patient as the time for a decision increases controls voluntary action in humans. Each time a proposal or objection reaches "Consciousness", its chance of becoming conscious again diminishes.
Avoidance of Goal Conflict	All goal conflicts are not informative and do not need conscious intervention. Goal conflicts can be avoided unconsciously in the instantiated hierarchical goal context via spreading of inhibitive activation energy.
Choice Points in Goal Hierarchy	Hierarchical Goal Contexts can have choice points and some decisions at choice points can be made unconsciously mainly based on situational motivation. Choice points in the action control bring uncertainty and conscious intervention, but not all.
Goal Hierarchy Instantiation	Goal hierarchy is instantiated either by a conscious event or unconscious process. The unconscious one happens via a priming effect from preattentive or subliminal perception process that senses the internal state and/or the environment.
Action Types	Three action types in a hierarchical goal context: consciously mediated, unconscious, and voluntary.
Drives as Self-Concept	Drives and their importance values are part of a self-concept as they are the deeper parts of Goal Context Hierarchies that shape experiences in self-monitoring.

IDA's domain is fairly complex and she requires *deliberation* in the sense of creating possible scenarios, partial plan of actions, and choosing between them

(Sloman 1999). In our example, IDA now has a list of a number of possible jobs in her workspace, together with their fitness values. She must construct a temporal scenario for at least a few of these possible billets to see if the timing will work out (say if the sailor can be aboard ship before a departure date). In each scenario the sailor leaves his/her current post during a certain time interval, spends a specific length of time on leave, possibly reports to a training facility on a certain date, uses travel time, and arrives the new billet with a given time frame. Such scenarios are valued on how well they fit the temporal constraints (the gap) and on moving and training costs. These scenarios are composed of scenes organized around events, and are constructed in the workspace by runs in the behavior net and "consciousness" modules as discussed before.

We humans most often select actions subconsciously. That is, with out conscious thought. But we also make voluntary choices of action, often as a result of the kind of deliberation described above. Baars argues that such voluntary choice is the same as a conscious choice (1997, p. 131). We must carefully distinguish between being conscious of the results of an action and consciously deciding to take that action, that is, consciously deliberating on the action. It is the later case that constitutes *voluntary action*. William James proposed the ideomotor theory of voluntary action (James 1890). James suggests that any idea (internal proposal) for an action that comes to mind (to "Consciousness") is acted upon unless it provokes some opposing idea or some counter proposal. GW theory adopts James' ideomotor theory as is (Baars 1988), and provides a functional architecture for it. The IDA model furnishes an underlying mechanism that implements the theory of volition and its architecture in a software agent (Franklin 2000; Kondadadi & Frnklin 2001).

Suppose that in our example at least one scenario has been successfully constructed in the workspace. The players in this decision making process include several proposing attention codelets and a timekeeper codelet. A proposing attention codelet's task is to propose that a certain job be offered to the sailor. Choosing a job to propose on the basis of the codelet's particular pattern of preference (different issues like priority, moving cost, gap, etc each with weight assigned to it). For example, our proposing attention codelet may place great weight on low moving cost, some weight on job fitness value, and little weights on others. This codelet may propose, say the second job on the scenario list because of its low cost and high fitness, in spite of low priority and sizable gap. If no other proposing attention codelet objects (by bringing itself to "consciousness" with an objecting message) and no other such codelet proposes a different job within a given span of time, the timekeeper codelet will mark the proposed as being one to be offered. If an objection or a new proposed job is made in a timely fashion, no job will be offered.

Two proposing attention codelets may alternatively propose the same two jobs several times. Several mechanisms tend to prevent continuing oscillation. Each time a codelet proposes the same job it does so with less activation and, so, has less chance of coming to "consciousness." Also, the timekeeper loses patience as the process continues, thereby diminishing the time span required for a decision. Finally, the metacognitive module watches the whole process and intervenes if things get too bad (Zhang et al. 1998a). This mechanism leads us to the voluntary action hypothesis in table 2.

Back to our example, once one or more jobs are voluntarily selected, IDA must generate an e-mail message offering them to the sailor. Due to her relatively narrow domain, IDA generates language (email messages) by filling in appropriate scripts. The scripts reside in behavior codelets and composed by *consciously mediated actions* of behavior stream(s). Thus, much of language production results from filling in blanks in learned scripts, and concatenating the results.

Differences with basic blackboard model

Here, we try to briefly discuss the difference between IDA' architecture and the basic blackboard model (Nii 1986). The basic blackboard model has three components: the *blackboard* or global datastructure, the *knowledge sources* or KSs, and the *control*. *KSs* are independent computational modules embodying specialized knowledge that is useful in solving a sub-problem in the context of a global application. KSs self-enabling and opportunistic and can communicate only though the blackboard. Control directs the overall computation of a system. The basic control cycle is (a) determine enabled KSs, (b) choose KSs for execution among the enabled, (d) execute the chosen KSs, which will cause a change in the blackboard, and (d) repeat the cycle. The functional parallel is that blackboard, KSs, and control respectively correspond to global workspace, contexts, and global broadcast (conscious event) in IDA. Functionally, both global broadcast and control are used to exercise global coordination.

We will point some of the differences. (1) Unlike KSs, contexts can communicate without global broadcasts. (2) Blackboard is a large working memory, while IDA's global workspace is a limited capacity. (3) *Control* picks the KSs that execute, but the global broadcast mostly influences the contexts except in the case of voluntary control. (4) Control chooses the KSs that access the blackboard while the conscious event chooses the content to be accessed by all. (5) More importantly, the basic blackboard model is a general architecture for computational systems while IDA is a cognitive agent architecture that implements the GW theory, and as such, it's a human like information-processing system, and is mostly constrained by cognitive and/or neurological evidences.

Issues related to the ASM

IDA's architecture in general and its ASM system in particular have been designed to be faithful to the GW theory. Table 1 shows some of the primary design assumptions or hypotheses. Table 2 focuses on the important design assumptions or hypotheses related to the ASM system or the concordance of the ASM to GW theory; in this section we briefly discuss about some of these.

Types of stream of actions

Actions are unconscious and they happen in a response to some external or internal change and they in turn cause changes in the system. In our mechanism, we can categorize actions into three types based on the relationship of the stream that control the actions to conscious experiences. That is, how a stream relates to a

conscious experience that precedes it and the conscious experience(s) during stream execution.

Consciously mediated stream of actions: For such actions, the controlling stream (dominant goal context hierarchy) is triggered by a conscious event via behavior-priming codelets but with out being explicitly conscious of the behaviors (goal contexts) in the stream. During the execution of such a stream, one or more its behavioral actions can cause informative changes to make it to "Consciousness" by some attention codelet. I.e., the effect of the actions becomes a conscious experience and the information is broadcasted. The broadcast information can activate behavior-priming codelets that can reinforce the already instantiated stream and bind and/or rebind some variable slots of the behaviors. More behavioral actions of the stream can cause more conscious experiences. This way, before an instantiated stream, as a goal hierarchy, achieves its stated goal, a number of conscious-unconscious-conscious (CUC) triads could happen. If a stream causes more than one CUC triad during its execution, then the actions of the steam are called *consciously mediated actions*. In general, action-induced conscious experiences can activate behavior-priming codelets that could instantiate new competing/cooperating stream(s).

Unconscious stream of actions: These types of actions are produced by streams instantiated in the same way as consciously-mediated actions but none of the actions of the behaviors in the stream trigger a conscious event. As a result, once the stream instantiated, it completes its execution unconsciously except possibly the last goal state achieved at the end of the stream execution; i.e., at most, only one CUC triad.

Voluntary stream of action: Here the instantiation of a behavior stream is triggered by a voluntary decision. That is, the goal of the stream is in "Consciousness" as a goal-image before the stream is instantiated in the behavior net. In other words, a stream is instantiated by its own goal-image. Voluntary actions are automatic and the stream execution will not be mediated by "Consciousness".

Through *experience* or *over-learning*, consciously mediated and voluntary streams of actions can become unconscious streams of action.

Characteristics of the ASM as a goal context hierarchy

Goal significance: If an instantiated stream has a behavior with a precondition (sub-goal) that has not been satisfied for some period of time, then the stream will be stuck and it cannot satisfy its goal(s). An attention codelet, with the motivation (activation) level or the significance of the stuck behavior, can try to bring the situation to "consciousness". The broadcast of the situation will allow recruiting of other streams that can resolve the sticking point. This means, the "consciousness" mechanism facilitates the process of building goal hierarchies (like the one in fig. 3) dynamically in the active behavior net system. GW theory states that a goal/sub-goal with enough significance is considered to be informative and may trigger a "conscious" intervention, so, a concordance with the theory.

Problem solving: The conscious-mediation is helpful to recruit a number of streams that cooperate under a goal hierarchy to create the unconscious process of problem solving. Each instantiated stream is an island of a partial solution and the conscious mediation (with its role for global coordination and control) helps to connect the different islands to reach to the complete solution.

Bias to Dominant Goal Hierarchy: A stream that has an executing behavior is an active action-plan. The behavior net algorithm has an implicit bias that gives momentum for the completion of an active plan over other competing plans that may exist. This is due to the fact that activation level of behaviors does not get reinitialized except for the behavior that was just active. That is, the past activation dynamics creates an activation momentum that biases the selection dynamics towards matured active plans; therefore, it is likely that the next active behavior to be from the active stream. This fits the GW theory in which Dominant Goal Hierarchies consisting of a coherent stream of goal contexts have a tendency to remain dominant until their goal is achieved. At any moment, but with less likely hood, any goal context from a competing goal context hierarchy can disrupt the dominant goal hierarchy.

Unconscious goal conflict resolution: The behavior net resolves goal conflicts at the goal and behavior nodes by spreading inhibitive activations. This hypothesizes (table 2) that some goal conflicts in goal context hierarchies can be resolved unconsciously in the ASM system; all goal conflicts are not novel.

Unconscious decision of choice points in goal hierarchies: Two or more behaviors or streams can satisfy a sub-goal of another stream. Satisfying this sub-goal creates a choice point; i.e., two chose one of the competing streams. The unconscious process of activation dynamics in the behavior net will make one choice point more relevant than the other one mainly based on a situational motivation; thus all choice points in goal hierarchies do not need conscious intervention.

Conclusion and closing remarks

The IDA in general and the ASM in particular have performed without any failure. After careful tuning of the global parameters, the ASM system produced the expected stream of actions as designed. One major reason for the absence of error, we think, is that IDA's domain does not allow for many competing streams to be instantiated for a given operating context. Although IDA has a fairly complex domain, it is not a proof of concept domain we hoped it to be.

This paper presented an overview of the "conscious" agent architecture, IDA, with a particular focus on the action selection mechanism (ASM), and how it integrates with the "consciousness" module. These two play a central role in the architecture, and in implementing global workspace theory. The action selection system is a collection of streams that comprise goals/intentions and behaviors. Instantiated streams along with drives constitute the hierarchical goal context of IDA.

Action selection is not solely the responsibility of the behavior net mechanism. Primarily, the action selection mechanism is highly influenced by the "consciousness" mechanism in two ways: (1) the conscious content influences the situational relevance of the behavior net, and (2) "consciousness" can impose voluntary action. Occasional metacognitive control and frequent emotional influences improve the action selection system. We believe that all these mechanisms, along with the ability to interleave unconscious actions and conscious

events, render our architecture suitable for information software agents that handle complex domains.

Instantiated streams do what is expected of a goal context hierarchy in GW theory. We pointed out the significant design assumptions or hypotheses in the different modules of IDA. We particularly discuss, in more detail, about the design assumptions or hypotheses in the action selection system and its concordance to the GW theory in its role as a goal context system. We hope that the design decisions made could be testable hypotheses and contribute to the furthering of the theory of cognitive modeling in general and the study of "consciousness" in particular.

Future work

Though the IDA architecture cuts a broad swath, human cognition is far too rich to be easily encompassed. Still, we plan to extend the model in several ways. Particularly, the behavior net is implemented in such a way that it can accommodate our ongoing research work that includes behavioral learning, creative problem solving, and the automation of action. Behavioral learning is done by adapting existing behavior streams, and needs an effective episodic memory (Negatu and Franklin 1999). The behavior net as a hierarchical goal context along with the "Consciousness" module has a significant contribution to incorporate non-routine problem solving capability into our agent architecture (Franklin 2001b). Action automation is also important to internalize skills from experience.

References

Agre, P. & Chapman, D. (1987). Pengi: An Implementation of a Theory of Activity. *Proceedings of Sixth National Conference on AI*, AAAI-87. Los Altos, CA: Morgan Kaufmann.

Anderson, J. R. (1990). *The Adaptive Character of Thought*. Hillsdale, NJ: Erlbaum.

Baars, B.J. (1988). *A Cognitive Theory of Consciousness*. Cambridge: Cambridge University Press.

Baars, B.J. (1997). *In the Theory of Consciousness*. Oxford: Oxford University Press.

Baars, B.J. (2002). The conscious access hypothesis: origins and recent evidence. *TRENDS in Cognitive Sciences*, Vol. 6 No. 1, (PP. 47-52). Jan. 2002.

Bogner, M., U. Ramamurthy, and S. Franklin. (1999). "Consciousness" and Conceptual Learning in a Socially Situated Agent. In *Human Cognition and Social Agent Technology*, Advances in Consciousness Research Series, 19. Ed. K. Dautenhahn. Amsterdam: John Benjamins.

Decugis, V. & Ferber, J. (1998). Action selection in an autonomous agent with a hierarchical distributed reactive planning architecture. *Proceedings of the second international conference on autonomous agents.* Minneapolis, MN USA.

Damasio, A. R. (1994). *Descartes' Error*. New York: Gosset; Putnam Press.

Dorer, Klaus. (1999). Behavior Networks for Continuous Domains using Situation-Dependent Motivations. In *International Joint Conference on Articial Intelligence* (IJCAI'99), pages 1233-1238.

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

Franklin, S. (1995). *Artificial Minds*. Cambridge MA: MIT Press.

Franklin, S. (1997). Autonomous Agents as Embodied AI. *Cybernetics and Systems* 28:499–520.

Franklin, S. (2000). Deliberation and Voluntary Action in 'Conscious' Software Agents. *Neural Network World* 10:505-521.

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

Franklin, S. (2001b). An Agent Architecture Potentially Capable of Robust Autonomy. *AAAI Spring Symposium on Robust Autonomy*, American Association for Artificial Intelligence; Stanford, CA; March.

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

Franklin, S. Kelemen, A. & McCauley, L.; (1998). IDA: A Cognitive Agent Architecture. *IEEE Conference on Systems, Man and Cybernetics*.

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

Holland, J. H. (1986). A Mathematical Framework for Studying Learning in Classifier Systems. In *Evolution, Games and Learning: Models for Adaption in Machine and Nature*, vol. al, Amsterdam, ed. D. Farmer. : North-Holland.

Jackson, J. V. (1987). Idea for a Mind. Siggart Newsletter, 181:23–26.

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

Kintsch, W. 1998. Comprehension. Cambridge: Cambridge University Press.

Kondadadi, R. & Franklin, S. (2001). A Framework of Deliberative Decision Making in "Conscious" software Agents. In *Proceedings Of Sixth International Symposium on Artificial Life and Robotics* (AROB-01).

Kokiniv, B. (1994a). A hybrid model of reasoning by analogy. In K. Holyoak and J. Barnden (Ed). *Advances in connectionist and neutral computation theory. Vol. 2: Analogical connections.* Norwood, NJ: Ablex.

Kokiniv, B. (1994b). The DUAL cognitive architecture: A hybrid multi-agent approach. *Proceedings of the Eleventh European Conference on AI.* Wiley.

Laird, E. J., Newell, A. & Rosenbloom, P. S. (1987). SOAR: An Architecture for General Intelligence. *Artificial Intelligence* 33:1–64.

Maes, P. (1990). How to do the right thing. *Connection Science* 1:3.

McCauley, T. L., & Franklin, S. (1998). An Architecture for Emotion. *AAAI Fall Symposium Emotional and Intelligent: The Tangled Knot of Cognition"*; AAAI; Orlando, FL.

McCauley, L., Franklin, S. & Bogner, M. (2000). An Emotion-Based "Conscious" Software Agent Architecture. In *Affective Interactions*, Lecture Notes on Artificial Intelligence ed., vol. 1814, ed. A. Paiva. Berlin: Springer.

Minsky. (1986). The Society of Mind. New York: Simon & Schuster.

Negatu, A. & Franklin, S. (1999). Behavioral Learning for Adaptive software Agent. *Intelligent Systems, Proceeding of the ISCA 8th International Conference*, pp. 91 – 95, Denver, Colorado.

Nii, H. P. (1986). "Blackboard Systems: The Blackboard Model of Problem Solving and the Evolution of Blackboard Architecture", *AI Maagazine 7 (2).*

Rhodes, B. (1995). Pronomes in Behavior Nets. *Technical Report # 95-01* . MIT Media Lab, MIT, Mass.

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.

Song, H., & Franklin, S. (2000). A Behavior Instantiation Agent Architecture. *Connection Science* 12:21-44.

Tyrell, T. (1993). Computational Mechanisms for Action Selection. *PhD Thesis*. University of Edinburg, UK.

Zhang, Z., Dasgupta, D., & Franklin, S. (1998). Metacognition in Software Agents using Classifier Systems. *In Proceedings of the Fifteenth National Conference on Artificial Intelligence*. Madison, Wisconsin.