# A Large-Scale Multi-Agent System for Navy Personnel Distribution

Lee McCauley and Stan Franklin

Institute for Intelligent Systems
The University of Memphis
373 Dunn Hall
Memphis, TN  38152

Office: (901) 678-2486
Fax: (901) 678-2380

{mccauley|franklin}@memphis.edu

**Abstract**

In the US Navy, at the end of each sailor's tour of duty, he or she is assigned to a new job. The Navy employs some 280 people, called detailers, full time to effect these new assignments. The IDA (Intelligent Distribution Agent) prototype was designed and built to automate, in a cognitively plausible manner, the job of the human detailers. That model is being redesigned to function as a multi-agent system. This is not a trivial matter due to the fact that there would need to be approximately 350,000 individual agents. There are also many issues relating to how the agents interact and how all entities involved, including humans, exercise their autonomy. This paper describes both the IDA prototype and the Multi-Agent IDA system being created from it. We will also discuss several of the major issues regarding the design, interaction, and autonomy of the various agents involved.

Keywords: multi-agent, large-scale, interaction, autonomy, agent-human roles

## Introduction

In the US 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 280 people, called detailers, full time to effect these new assignments. Each detailer serves a community of sailors distinguished by paygrade and job skills. IDA (Intelligent Distribution Agent) is a “conscious” software agent developed for the US Navy (Franklin et al. 1998) to facilitate the distribution process by playing the role of detailer. She must communicate with sailors via email using natural language, understanding the content and producing human-like responses. 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. In doing so she must adhere to some ninety policies. She must hold down moving costs, and she must cater to the needs and desires of the sailor as well as is possible. This includes negotiating with the sailor via an email correspondence in natural language. IDA’s mechanisms will be briefly described in the next section with references to detailed accounts.

The IDA prototype was designed and implemented around the primary goal of cognitive modeling. That is, she was intended to not only perform the functions of a human detailer, but to perform them in a way that conforms in various respects to several cognitive models. The end result is a system that is cognitively plausible, fulfils the routine tasks for a human detailer, and is highly inefficient from a computer science standpoint. In part to remedy this later issue, the technology developed for the IDA prototype is being redesigned to function as a multi-agent system. The latter part of this paper will focus on how the IDA model is being subdivided into separate agents responsible for representing their human counterparts. We will also describe the interactions between the agents, design issues, and the issues surrounding agent autonomy.

## The IDA Architecture

The IDA architecture, shown in figure 1, consists of both an abstract level (containing such entities as behaviors, message type nodes, etc.), and a lower, more specific level (implemented by small pieces of code). At the higher level the architecture is quite modular with module names often borrowed from psychology. There are modules for Perception, Action Selection, Associative Memory, Emotions, Constraint Satisfaction, Language Generation,

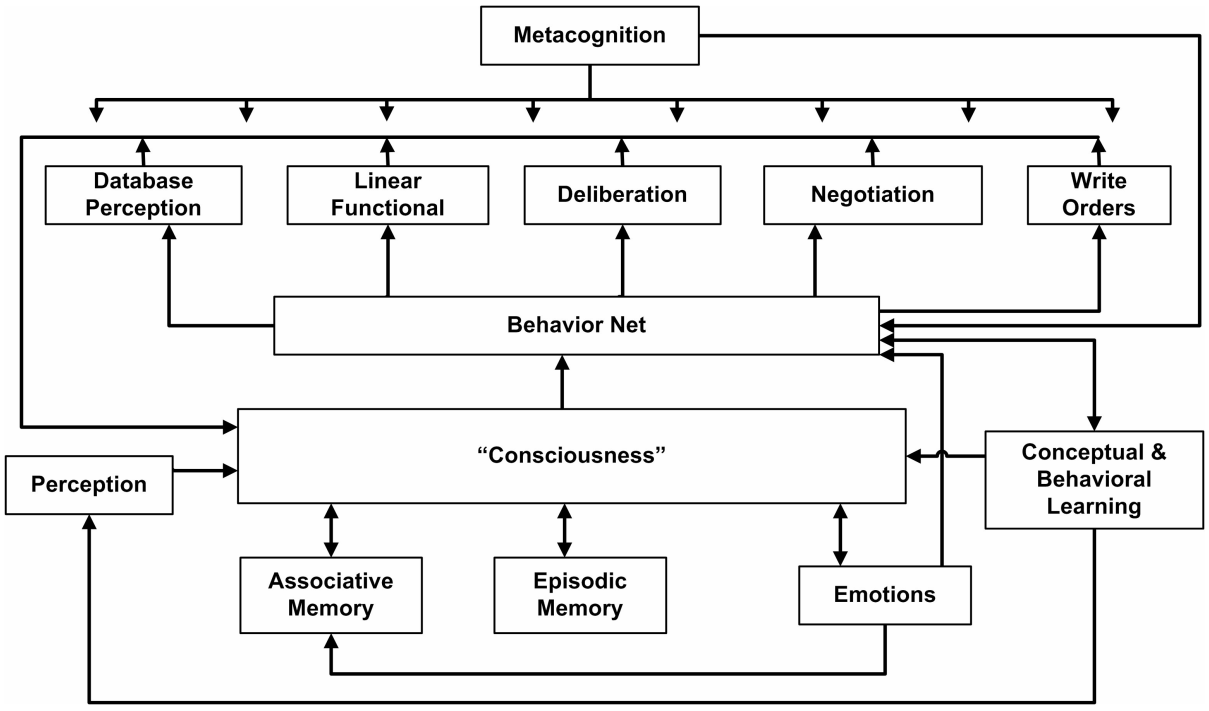

Figure 1: The IDA Architecture

Deliberation, and "Consciousness". Many of their mechanisms were inspired by ideas from the "new AI", a recent reincarnation of artificial intelligence that's less symbolic, more connectionist, and more biologically motivated (the copycat architecture (Mitchell 1993, Hofstadter & Mitchell 1994, Sloman 1999), behavior nets (Maes 1989), sparse distributed memory (Kanerva 1988), and pandemonium theory (Jackson 1987).

In the lower level of the IDA architecture the processors are all small, dumb pieces of code called codelets. Specialized for some simple task, they often play the role of demon waiting for the appropriate condition under which to act. Most of these codelets subserve some high level entity such as a behavior or a slipnet node. Some codelets work on their own, performing such tasks as watching for incoming email and instantiating goal structures. An important type of the latter is the attention codelets who serve to bring information to "consciousness". Codelets do almost all the work, making IDA a multi-agent system in the sense expressed by Minsky (1985).

In the rest of this section we briefly describe each of IDA's major cognitive modules giving references to detailed accounts. We begin with how she perceives.

## Perception

IDA senses strings of characters, not imbued with meaning, but as primitive sensation as, for example, the pattern of activity on the rods and cones of the retina. This text may come from email messages, a chat room environment, or from a database record. Her perception module employs analysis of surface features for natural language understanding (Allen 1995). It partially implements perceptual symbol system theory (Barsalou 1999). Its underlying mechanism constitutes a portion of the Copycat architecture (Hofstadter & Mitchell 1994). The perceptual/conceptual knowledge base of IDA takes the form of a semantic net with activation passing called the slipnet (see figure 2). Nodes of the slipnet constitute the agent’s perceptual symbols. Pieces of the slipnet containing nodes and links, together with codelets whose task it is to copy the piece to working memory constitute Barsalou’s perceptual symbol simulators. There's a horde of perceptual codelets that descend on an incoming message, looking for words or phrases they recognize. When such are found, appropriate nodes in the slipnet are activated. This activation passes around the net until it settles. An idea type node (or several) is selected by its high activation, and the appropriate template(s) is filled by codelets with selected items from the message. The information thus created from the incoming message (Franklin 1995) is then written to the perception registers in the workspace (to be described below), making it available to the rest of the system. Almost all IDA's modules either write to the workspace, read from it, or both; it constitutes her short-term memory.

## Associative Memory

IDA employs sparse distributed memory (SDM) as her major 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. How does IDA use this associative memory? Reads and writes to and from associative memory are accomplished through a gateway with the workspace called the focus. When any item is written to the workspace, another copy is written to the read registers of the focus. The contents of these read registers of the focus are then used as an address to query associative memory. The results of this query, that is, whatever IDA associates with this incoming information, are written into their own registers in the focus (see figure 3). This may include some emotion and some action previously taken. Thus associations with any incoming information, either from the outside world, or from some part of IDA herself, are immediately available. Writes to associative memory are made at several key points based on the type of response that IDA is formulating.

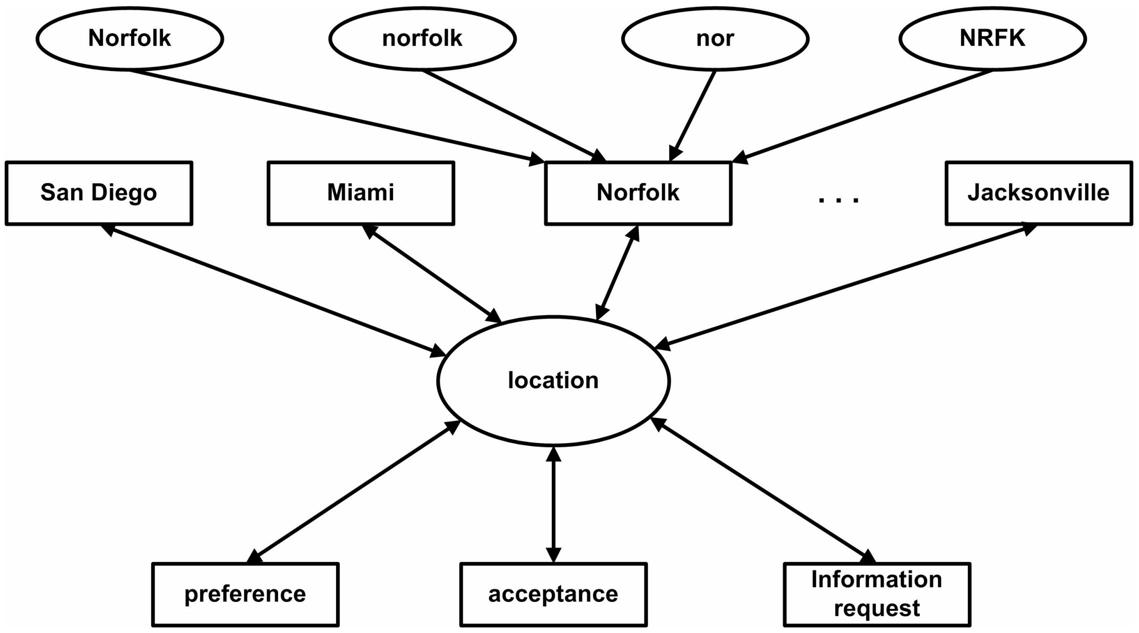

Figure 2: A portion of IDA's slipnet

## "Consciousness"

Implementing parts of global workspace theory (Baars 1988,1997), the apparatus for producing "consciousness" consists of a coalition manager, a spotlight controller, a broadcast manager, and a collection of attention codelets who recognize novel or problematic situations (Bogner 1999, Bogner et al. 2000). Attention codelets have the task of bringing information to "consciousness". Each attention codelet keeps a watchful eye out for some particular situation to occur that might call for "conscious" intervention. Upon encountering such a situation, the appropriate attention codelet will be associated with the small number of 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 its coalition might compete for "consciousness". In IDA the coalition manager is responsible for forming and tracking coalitions of codelets. Such coalitions are initiated on the basis of the mutual associations between the member codelets. At any given time, one of these coalitions finds it way to "consciousness", chosen by the spotlight controller, who picks the coalition with the highest average activation among its member codelets. Global workspace theory calls for the contents of "consciousness" to be broadcast to each of the codelets. The broadcast manager accomplishes this.

## Action Selection

IDA depends on an expansion of the idea of a behavior net (Maes 1989) for high-level action selection in the service of built-in drives. 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 codelets for their execution. A behavior net is composed of behaviors and their various links. A behavior looks very much like a production rule, having preconditions as well as additions and deletions.  A behavior is distinguished from a production rule by the presence of activation, a number intended to measure the behavior’s relevance to both the current environment (external and internal) and its ability to help satisfy the various drives it serves. Each behavior occupies a node in a digraph.

As in connectionist models, this digraph spreads activation. The activation comes from activation stored in the behaviors themselves, from the external environment, from drives, and from internal states. The more relevant a behavior is to the current situation, the more activation it's going to receive from the environment. Each drive awards activation to every behavior that, by being active, will help to satisfy that drive. Certain internal states of the agent can also send activation to the behavior net. This activation, for example, might come from a coalition of codelets responding to a “conscious” broadcast. Finally, activation spreads from behavior to behavior along links. To be acted upon a behavior must have its preconditions satisfied, activation over threshold, and must have the highest such activation. Behavior nets produce flexible, tunable action selection for these agents.

IDA’s behavior net acts in concert with her “consciousness” mechanism to select actions. Suppose some piece of information is written to the workspace by perception or otherwise. Vigilant attention codelets watch both it and the resulting associations. One of these attention codelets may decide that this information should be acted upon. This codelet would then attempt to take the information to “consciousness”, perhaps along with any discrepancies it may find with the help of associations. The attention codelet and the needed information carrying codelets become active. If the attempt is successful, the coalition manager makes a coalition of them, the spotlight controller eventually selects that coalition, and the contents of the coalition are broadcast to all the codelets.  In response to the broadcast, appropriate behavior priming codelets perform three tasks: (1) if not already there, an appropriate goal structure is instantiated in the behavior net, (2) wherever possible the codelets bind variables in its behaviors, and (3) the codelets send activation to its most relevant behavior. That behavior may be chosen, eventually, to be acted upon.

The rest of the behavior codelets associated with the chosen behavior then perform their tasks. An action has been selected and carried out by means of collaboration between "consciousness" and the behavior net.

## Constraint Satisfaction

At the heart of IDA's task of finding new jobs for sailors lies the issue of constraint satisfaction. Not only must IDA look out for the needs of the sailor, she must also see that the requirements for individual jobs are met, and simultaneously adhere to the policies of the Navy. Taking such issues into consideration, IDA's constraint satisfaction module is designed to provide a numerical measure of the fitness of a particular job for a particular sailor.

Given a specified issue such as sailor preference, a particular Navy policy or specific job requirement, referred to as $j$ for short, we define a function $x_j$ that provides a numerical measure of the fitness of this job for this sailor with respect to this particular issue. For example, suppose the issue $j$ is the one that says a sailor may be assigned to a job requiring a certain paygrade, if his or her paygrade is no more than one more or less. Here we might define $x_j$ as follows: $x_j = 1$ if the sailor has the specified paygrade, $x_j = 0.5$ if the sailor's paygrade is one more or less than that specified, and $x_j = 0$ otherwise. This would provide the desired numerical measure of fitness with respect to this particular policy. We also need, for each issue $j$, a numerical measure $a_j$ of the relative importance of that issue with respect to all the other issues. Such measures can be determined in consultation with expert human detailers using statistical methods. They may also be approximated from data concerning actual assignments of sailors to jobs by human detailers. Each $a_j$ will be used to weight the result of the corresponding function $x_j$ that measures the suitability of the given job for the sailor in question with respect to the issue j. Thus the weighted sum of the $x_j$, $\Sigma a_j x_j$, will give the required total fitness measure with respect to all the issues.

## Deliberation

IDA's relatively complex domain requires deliberation in the sense of creating possible scenarios, partial plans of actions, and choosing between them (Franklin 2000, Kondadadi & Franklin 2001). For example, suppose IDA is considering a sailor and several possible jobs, all seemingly suitable. She must construct a scenario for each of these

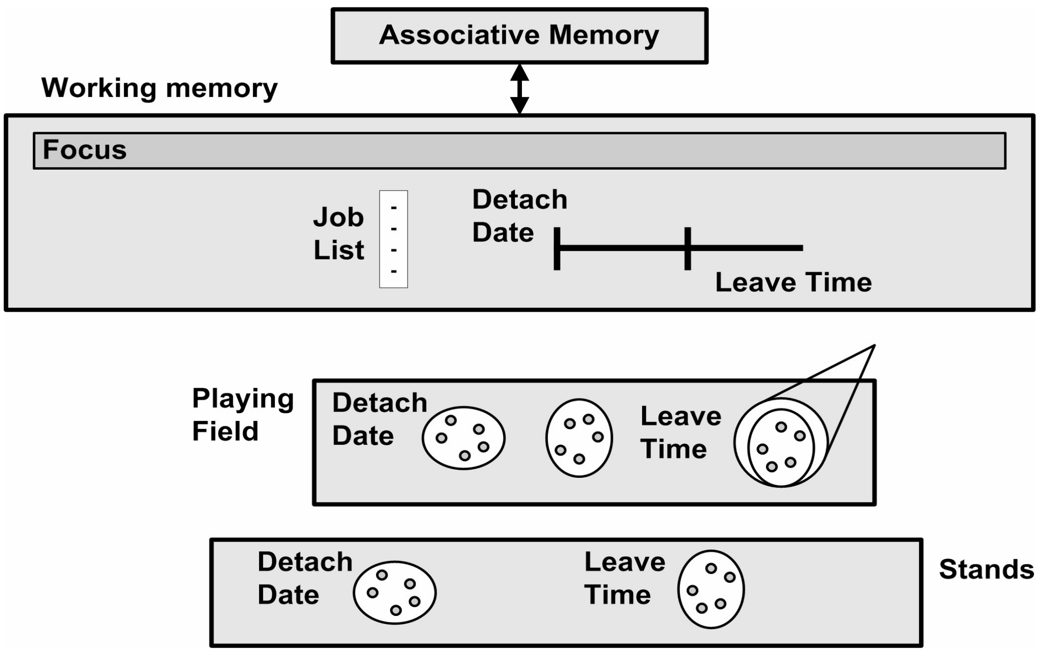

Figure 3: IDA's deliberation in action

possible billets in order to determine whether or not a given job can meet joint temporal constraints such as the sailor's projected rotation date (PRD) and the take-up month (TUM) of the billet. And, a sailor to be assigned to a certain ship had best arrive before the ship sails. In each scenario the sailor leaves his or her current position during a certain time interval, spends a specified length of time on leave, possibly reports to a training facility on a certain date, and arrives at the new billet with in a given time frame. There's travel time to be figured in. Such scenarios are valued on how well they fit these temporal constraints as well as on moving and training costs.

At this point in IDA's search for a job for the given sailor, a list of jobs coarsely selected from the current requisition list is already in the workspace. One by one they've been acted upon by the constraint satisfaction module resulting in an attached numerical fitness value. Some attention codelet notices that the last fitness value has been written next to its job. This is its cue to begin the scenario building process. An attention codelet selects a job for the scenario (typically the one with the highest fitness) and recruits information codelets to carry specific information about the job.

This process continues over and over again writing a scene for travel time, for the beginning of a training class (if needed), for the time interval of the class (if needed), and for the report-no-later-than date. After the last scene is written IDA computes the gap, the difference between the report date and the take up month. If the former is within the later, the gap is zero, otherwise more. If the gap is non-zero, the adjust-the-gap attention codelet will try to instigate the building of a new scenario for the same job with a different detach date that may produce a smaller gap. It is also possible that an attention codelet may like this job and want to propose that it be one of those offered to the sailor or, conversely, an attention codelet may dislike this job and suggest that it be removed from consideration.

## Implications for Autonomy

Is IDA autonomous and, if so, what does this say about autonomy in general? Franklin and Graesser (1997) define an autonomous agent as a system situated in, and part of, an environment, which senses that environment, and acts on it, over time, in pursuit of its own agenda. Is IDA autonomous by this definition? Let's check the conditions. IDA's environment consists of her community of sailors, the various US Navy databases that she accesses, and the piece of the Internet including her email client and email service provider. IDA senses her environment via character strings from incoming email messages and from database queries. She acts on it by sending queries (active perception) and email messages to sailors. She does so over time and, as we shall argue below, in pursuit of her own agenda. Further, the messages she sends certainly influence the responses she receives from sailors in her community. She is structurally coupled to her environment.

The essence of autonomy in the definition of autonomous agent seems to be this notion of the agent pursuing its own agenda (see also Castelfranchi 1995). A non-autonomous or less autonomous agent might well be expected to sense its environment and act on it, and to be structurally coupled. Such an agent would depend on a human (or other autonomous agent) as a 'user' to tell it what to do and, perhaps, how to do it. Does IDA have a user?

If IDA has a user, or users, they would have to be the sailors in her community since these are the only humans (autonomous agents) with whom she has contact while she is running. Does a sailor in her community tell IDA what to do or how to do it? Suppose a sailor sends IDA a 'find me a job' message. Does that constitute a command? We think not, since IDA would in time offer that sailor possible new jobs on her own volition even if she hadn't received his or her message. She periodically checks the rollover list for her community and writes to sailors approaching their PRD (projected rotation date). This behavior is motivated by her built-in drives and by the date.

Let's ask the question in a different way. Is IDA an agent of the sailor in the sense of a personal assistant or that of an entertainer's agent? Again we think not. IDA offers jobs to the sailor and tries to convince the sailor to accept one of them. However, on occasion she will assign a job to a sailor over his or her objection, something no assistant or entertainer's agent could or would do. Recall that IDA automates the tasks of a human detailer. Is a sailor a user of his or her human detailer? Sailors certainly don't think so. They are almost universally suspicious that detailers save the best jobs for their friends.

How about telling IDA how to do it? Suppose a sailor writes asking IDA for a particular job? Does that constitute telling IDA both what to do and how to do it? Again, IDA seems to be autonomous. She may decide to give that job to the sailor, or she may refuse. It's her choice. She's autonomous.

Another question that's often raised is whether the autonomy truly belongs to the software agent or to its designer. We've designed IDA to have the motives (drives) that so influence her actions. Aren't they really the designer's motives, and not IDA's? We grant that the designer originally provides the motives. But, once the agent is complete and running, the motives are those of the agent, and cannot be further affected by the designer. The motives are then those of IDA. But what if we redesign IDA with different motives? In that case we now have a different agent. An analogy may help. A human's hunger drive and sexual drive has been evolved-in, but are now among the motives of that human, not of evolution.

Another issue is the apparent trade-off between autonomy and robustness, the ability to perform well under unexpected circumstances. When autonomy increases, must robustness decrease and vice versa? It certainly seems that the more independence (autonomy) we grant a software agent, the less predictable its actions may become especially in a dynamic and unpredictable environment. Might it not be less robust as a consequence? What's the situation with IDA? Her domain, while certainly dynamic, is relatively predictable. As we'll argue below, IDA typically handles routine problems with novel input. Within this environment she has proved quite robust. We conclude that robustness and autonomy need not always trade-off. In some circumstances we can have both. But what about in less predictable environments?

Robust autonomy on the part of software agents requires, at least in part, the ability to deal appropriately with novel and unexpected situations. According to global workspace theory, dealing with such situations is one of the primary functions of consciousness in humans. Though her "consciousness" module is designed to deal intelligently with novel, unexpected, and problematic situations, IDA is expected to deal only with novel instances of routine situations. One 'find me a job' message from a sailor is much like another in form, even in natural language with no agreed upon protocol. Similarly, finding a new billet for one sailor will generally require much the same process as for another. Even the negotiation process between IDA and a sailor turns out to be relatively routine. From analysis of a corpus of messages we've constructed a complex, but quite finite, flow chart of possible messages types and responses.

However, we expect IDA to occasionally receive messages outside of this expected group. Can she handle such a message intelligently by virtue of her "consciousness" mechanism alone? We doubt it. Some attention codelet will be needed to bring the novel message to "consciousness". Some behavior priming codelets will be needed to instantiate an appropriate behavior stream (goal hierarchy) needed to deal with the situation (Franklin 2001). Perhaps a single, novel-situation attention codelet will respond to a percept by default if no other attention codelet does so within its time interval. This novel-situation attention codelet would try to bring information about the novel situation to "consciousness". The broadcast would, hopefully, recruit behavior priming codelets to instantiate a behavior stream able to cope with the situation. Suppose there is no such stream? Well, we humans can't cope with every situation either, but we try. And, we combine goal hierarchies in novel ways. This combining ability would seem a necessary ingredient if a "conscious" software agent were to be truly robustly autonomous. It also seems that learning must play a role here.

We would conclude that "conscious" software agents present a promising architecture and collection of mechanisms from which to start in trying to design truly robust autonomous agents. But, clearly, there's lots of work to be done, particularly about the handling of non-routine problems.

## Multi-Agent IDA

The IDA prototype was not intended to be field deployable; it was intended to explore the ways and the degree to which the human detailer's function could be automated. The knowledge and technology acquired during the IDA project is now being applied to a system that goes beyond the current Navy processes. The goal of this new system is to improve the satisfaction of the sailors with the assignments they receive, the satisfaction of the commands with the sailors that they are assigned, and the utilization of other resources and factors (i.e. movement costs, time a job is left unfilled, etc.).

Part of improving the satisfaction of the sailors involves providing them with the feeling that their needs are being taken into consideration while jobs are being handed out. Each sailor will, therefore, have a software agent whose purpose is to represent the sailor, including rating jobs based on the sailor's needs, abilities, and stated desires, and negotiating for desirable jobs. This Sailor Agent (SA) will not care about general Navy issues such as fleet balance or manning ratios, and it will not care about command issues such as billet gap (time of job vacancy). For this reason, the sailor can be confident that their SA will be looking out for his or her needs above all else.

Similarly, the commands, which have had little to no say in their incoming sailors, should also be represented in a way that lets them influence who is assigned to their jobs. For this, each command will have a software agent representative whose purpose is to rate sailors based on a job's requirements and the command's stated desires for that job, and to negotiate with desirable sailors. As with the SA, the Command Agent (CA) will not care about a sailor's location preference or general Navy fleet balance.

Of course, the Navy as a whole must also have some say in how jobs are assigned in order to maintain overall mission readiness. Neither the SAs nor the CAs will take Navy concerns into account when making their judgments. This task falls upon the Navy Agent (NA). Even though there will be a SA for each sailor and a CA for each command their will be only one NA. In implementation there may actually be several incarnations of the NA for efficiency purposes, but they will act as one agent. While the exact responsibilities of the NA have not yet been identified they are likely to include the creation of a population of manning scenarios ranked by their adherence to Navy policies and desires, and the doling out of incentive credits to the commands that can be used to affect a given sailor's acceptance of a job. The incentive system is a way for the Navy to influence the job assignments toward its goals while still allowing the commands and the sailors to have maximum input into their own situations. The incentive credits may be in the form of additional salary dollars, additional leave time, or any number of other as of yet unspecified enticements that can be given to a command based on the ranking of its jobs, the historical difficulty of filling its jobs, etc. Some of the incentives will likely be attached to specific jobs while some portion of them will be usable at the command's discretion. This would allow, for example, a command to save incentives for use in luring a particularly well-qualified candidate to one of their jobs as opposed to a similar job in a historically more desirable location.

The rest of this section describes the issues that arise when considering the design of such a system and the issues relating to how the agents interact.

## Issues in Agent Design

The magnitude of the design issues come into perspective when one begins to calculate the number of distinct agents needed for such a system. There are currently over 300,000 sailors in the US Navy, each of whom would have their own SA to represent them. Likewise, there are approximately 45,000 commands that would need their own CA. Keep in mind that each of these agents needs to have most of the capabilities of the full IDA prototype although most of these abilities have been pared down to a smaller scope.

The most obvious question that must immediately be asked is, 'how can so many complex agents function together'. Even though there is no definitive answer for this, as of yet, there are a few aspects that might provide a glimmer of hope. First of all, each sailor only changes jobs once every three years, on average. Of the 100,000 sailors seeking new jobs in a year each one usually takes less than two weeks of negotiation with a detailer find a new position. This means that there will be roughly 2000 sailors seeking jobs during any two-week period. A similar estimate regarding the number of commands attempting to fill positions for a comparable period yields about 3000. This reduces the agents that need to be active at one time to just a few thousand. While this is still not a trivial task, it is at least tractable.

The first design decision made to assist with this issue involved the separation of the agents' (SAs and CAs) main functions and their differentiating data. In other words, everything that makes each SA or CA specific to its sailor or command is explicitly separate from the portions of the agent that perform non-specific functions. This element further reduces the number of active agents at any one time by allowing the specific data portion of the agent to be stored in an object database and then completely removing the agent from the system when it is not needed. When a message is sent to an agent that is not currently instantiated, that agent is revived with its specific data intact and given the message to be processed.

Since the Multi-Agent IDA system is not explicitly concerned with cognitive modeling, we also made several design decisions based on the need to increase the efficiency of the agents. We determined that, in IDA, the majority of our resources were being consumed in three areas: associative memory, behavior net, and "consciousness". We determined that it might be possible to replace the behavior net with a more efficient but less cognitively valid mechanism that we call "action managers". Essentially, an action manager is just an execution engine for "action calls". You could think of an action call as a reference to an action along with the parameters necessary to execute that action. The result of this is a simplistic proprietary programming language that is internal to the agent. Such a language allows human designers to use an external tool to describe the behaviors of the agent. It also provides the possibility, in the future, for the agents to know about their own actions and to create their own action manager to accomplish a given task. An additional benefit of this construction is that the actual actions can be shared by multiple agents and could even be executed on a remote system. At the moment, the other two elements, associative memory and “consciousness”, have been removed. Although IDA has a complex structure that should be able to handle complex tasks, the function that she actually performs is routine and, consequently, does not exercise the associative memory or “consciousness” modules to their full potential. For this reason, it was decided to attempt to perform these routine tasks using more focused mechanisms.

Another possibility for increasing efficiency would be to allow certain agent competencies, such as email perception, to be shared on a remote system in much the same way as was described previously for actions. This option, however, is not as straightforward as it might seem at first glance. A particular agent’s perception may be effected by its current context, which may consist of the messages and ideas that it has recently processed or its emotional state at the time just to name a couple. To farm this kind of ability out to an external system implies that the context can be formalized and presented to the perception mechanism. It is likely that as agents become less formalized, the possibility of sharing competencies decreases.

## Issues in Agent Interaction

Each agent in the system is pursuing its own agenda and attempting to influence the behavior of other agents in order to render the result it wants. These attempts at manipulation can range from subtle to blatantly obvious. Either way, to do this effectively the agents need to have some knowledge of the other agents involved, their relationships, and their motivations. This section describes each of the agents, which other agents they interact with, what their primary motivations are, and how they attempt to influence the other agents.

### Human Agents

There are three different categories of human agents, each of which has a corresponding software agent that, to some degree, is their proxy in the virtual job marketplace. Although the details of how each human interacts with its agent will be left to the sections describing each agent type, there are some commonalities in the mode of exchange. In particular, there are two methods that any of the human agents can use to communicate with their respective agents.

One way that a human interacts with its agent is through a web interface. A sailor, for example, would sit down at a computer system, go to the appropriate URL, and login to a web interface that is tailored to them. In addition to the standard web portal type information, such as weather reports or sports scores, the sailor would have access to his or her software agent that can assist them in finding a job that matches their needs and desires. The web server, in this case, is responsible for delivering messages from the human sailor to their software agent and vice versa. From a conceptual perspective, the web interface is an extension of the human's software agent. Each of the human agents would login to a web site that is specific to their role.

The other communication possibility is through email. This type of interface will likely be used only by those who do not have consistent or ready access to a web client, such as a sailor stationed aboard a submarine. Email perception in the Multi-agent IDA is essentially the same as in the IDA prototype although a slightly different set of ideas needs to be recognized due to the nature of the additional agents and their roles.

**Sailor Agents**

The SA has the purpose of recommending appropriate jobs to their human sailor that matches the sailor's capabilities and desires, and to facilitate the application for the jobs that the sailor wants; its primary goal is to make the sailor happy. As a starting point, the agent uses the sailor's personnel data from a Navy database to gather information relevant to its function. This would include the sailor's rank, past trainings, performance evaluations, whether the sailor has a spouse in the navy, how many dependents the sailor has, etc.

Even as extensive as the information is, it does not fully capture a sailor's desires or even relevant abilities. Through either the web interface or email, the SA will perceive information from the sailor such as preferences for a particular location or job type. As with IDA, these preferences can be fairly vague, such as a preference to be based in Virginia. A given job may also request certain experience or abilities that are not captured in the personnel data (e.g. a Master Chief that has been through a complete ship's overhaul). A CA would send these job specific requests to the SA, which would then ask the sailor for the needed information if it did not already have it. While it may not be the case, there is the implicit assumption that there is mutual trust between the sailor and his or her SA. This means that the SA assumes that anything that the sailor tells it is the truth and that the sailor trusts the SA to give him or her all possible options with respect to perspective jobs.

The SA must also interact with CAs. This can take the form of simple expressions of interest in a job represented by a CA all the way to complex negotiation interchanges involving incentive requests or responses. In this situation, it is important to note that neither bargaining agent has complete knowledge of the other agent's circumstances. Here, the SA does not know what incentives the CA has available, and the CA does not know what preferences the sailor has expressed to the SA.

The SA has limited direct interaction with the NA. The only time that a SA would get a message from the NA would be when the NA wants to highlight a particular job for which the sailor has not yet applied.

**Command Agents**

A CA is tasked with finding sailors to fill its command's jobs. Just as with the SA, the CA's primary purpose is to make the command happy. Some information about the jobs within a command is available to the CA through Navy databases. In addition, the command interfaces with its CA through a web interface (or email) that lets it describe non-standard skills or experience that it would like to have for a given job. It can also describe aspects of the job or command that would not be available to SAs under normal circumstances. The command might also give the CA a certain amount of incentive credits that it can use to entice sailors to accept a job.

The interaction of a CA with a SA could begin with the SA sending the CA a message asking whether its sailor would be considered for a job or it could begin with the CA sending a message to a SA requesting that its sailor apply for one of the CA's jobs. In this way, both the CA and SA can be proactive in seeking applicants or jobs respectively.

The CA also has limited direct interaction with the NA. A CA might get a message from the NA would be when the NA wants to bring a particular sailor to the attention of a CA that has not yet expressed an interest in that sailor for the job in question. The NA would also be the entity that would inform the CAs of the incentive credits that their command has to bargain with. Note that this is not the amount of incentive credits given to a CA to fill a given job; only the command can impart that to a CA.

**Navy Agent**

The NA plays a large indirect role in the selection process, but does not often interact with the other software agents directly. The human counterpart to the NA will probably be an organization within the Navy whose purpose is to maintain personnel mission readiness. That organization will be responsible for identifying overall trends that might affect general Navy desires such as fleet balance or sea billet under-manning. This information would then be used to come up with a strategy for the dispersal of incentive credits. It will also decide on what issues the Navy, as a whole, should be concerned with and how important each is.

Before the SAs or CAs get the job lists, the NA uses a multiple matching algorithm based on its input that creates a list of all possible job-to-sailor matches that would be acceptable to the Navy (not violate any policies) and scores each vector (or possibly match). When a SA accesses the job list, it will only see those jobs for which the NA has decided its sailor could apply. Likewise, when a CA accesses the personnel list, it will only see those sailors that the NA has deemed acceptable for the given job. Keep in mind that this is not a restrictive list; the SA still has access to all of the possible jobs for its sailor and the CA still has access to all of the possible sailors for its jobs. The purpose of this is threefold: (1) the Navy organization has the ability to change policies and the change will go into effect immediately, (2) the NA does not need to "bless" any matches to make sure that Navy policy is not violated, and (3) the SAs and CAs do not have to process a large number of jobs or sailors that could never match.

The NA's other two functions also derive from this list of job-to-sailor matches. The list could be the basis upon which some incentive credits are given to the commands. On the rare occasions when the NA directly interacts with a SA or CA, it does so in order to inform the other agents of possible matches that they have either overlooked or discounted for one reason or the other. For a SA, such a message from the NA means that there may be some additional incentives attached to that job if its sailor applies for it. For a CA, such a message from the NA means that the command may not have to expend as much of its own incentive credits to get a decent person in that job. This methodology almost guarantees that the best matches for the Navy are, at least, considered in every case. This does not, by any means, impose the Navy's desires on the sailors or the commands, but it will increase the likelihood that the Navy's concerns are met.

## Issues in Agent Autonomy

Unlike in the IDA prototype, the nature of autonomy in this system is a more complex and dynamic interplay between the agents involved. Each agent, while ultimately cooperating with the other agents, performs its actions based on utilitarian purposes (Brainov and Sandholm 1999). It is almost ironic that the solution to giving sailors and commands greater autonomy in the assignment process is to delegate some of that autonomy to software agents. The reason for this, of course, is that the process is complex enough that a human that does not do this for a living would not necessarily be able to adequately take all factors into account when judging the fitness of a job-to-sailor match. The agents provide assistance to their human counterparts by pre-qualifying and presenting the data in a way that reduces the volume of information needed to make an appropriate decision. The agents cannot, however, make all decisions. This is not a statement about the ability of the agents, but about the balance of autonomy necessary to maintain the goals of the system. For this reason, the autonomy of the agents in the Multi-agent IDA model must be both broad and restrictive. The purpose of the software agents would be compromised if they did not have the ability to make at least some decisions on their own. At the same time, some decisions must be left to the human agents, not because the SAs or CAs couldn't perform that function, but because it is more important that the humans involved feel satisfied with the results than that the system produces an "optimal" solution.

It is important to note that the use of the word "autonomous" here is meant to convey the degree to which an agent can act without perturbation from some other agent or entity (Hexmoor 2000; Mele 1995, 1992; Steels 1995). The agents involved in this system are autonomous to varying degrees based on the situations within which they are given the option of acting or deciding (Castelfranchi 2000). To look at the mix of decisions that the agents need to make we will describe the process flow that occurs during the course of a sailor finding a job. The process can be instigated from either the sailor side or the command side. Although the process is usually initiated by the human agents (either sailor or command), the SAs or CAs can begin the procedure preemptively if they feel that their humans are behind schedule (e.g. they haven't logged in in a timely manner) or if an opportunity presents itself (e.g. a particularly well suited sailor enters the marketplace).

If the sailor or SA is the instigating entity, then the process begins with the SA searching through the available jobs and rating them based on what it knows about the sailor. If there are any possible jobs available, then it sends a message to the sailor letting him or her know that it is starting the process. It then sends a message to all of the CAs whose jobs the SA had rated above a certain threshold. This message is simply a question to the CA as to whether the SA's sailor would be considered for that particular job. Note that at this point the SA has had the prerogative as to which CAs to contact. The CA receiving the message decides, on its own, whether the sailor in question is a close enough match for the job that they would be considered if an application was received. The answer to the SA's question is then returned. The SA keeps a running list of the responses so that it can present these to the sailor if requested through the web interface or email. At the point where the SA has received an answer for all of the questions sent out, it will send a message to the sailor relating this. The sailor then receives a list of all of the jobs that he or she might qualify for with the jobs ranked and those for which the SA has already contacted the CAs highlighted in some way.  This gives the sailor the ability to override the rankings of the SA.  It also gives the sailor an opportunity to ask the SA about certain jobs and explanations for the rankings.  The information requested of the SA may be available in a Navy database that the SA can readily access or it may need to ask for that information from the appropriate CA.  Any of these questions that the CA cannot answer would be asked of its human counterpart with the results sent back to the SA and, conversely, to the sailor.  When the sailor is ready, he or she tells its SA which jobs he or she wants to apply for along with some guidelines related to the sailor's desires for incentive credits attached to that job.  This could include a range that would allow the SA to do some of the bargaining as the sailor's proxy.  The SA then sends the applications to the right CA.  In a similar way to how the SA presented jobs to its sailor, the CA presents to its human a complete list of possible sailors that might fill a given job along with its rankings and the applications of any of the sailors on the list that it has received.  The command can, at this point, override the CA's rankings, ask questions about the sailors, tell the CA to contact a SA whose sailor has not yet provided an application, counter bid on a sailor's incentive request, reject an application, or accept an application thereby filling the position.  Just as the sailor did with its SA, the command would also have the opportunity to give the CA some specified latitude in the bidding process.

The other possible way that the interactions might be structured would involve the sailor applying for a number of jobs and providing their SA with a preference order.  The command, for its part, would provide a preference order to those sailors who applied, but would not have the ability to make a final selection.  Under this version, the final selection would be made by the NA, which would generate a best-case match from the Navy's perspective while taking the preference rankings of the sailors and commands into account.  The NA would not be able to assign a sailor to a billet for which they did not apply or give a command a sailor that they had not put in their "acceptable" list by virtue of assigning them a preference.  This does not mean that the sailors or the commands are going to, necessarily, get their first choice.  On the other hand, it is highly likely that, if a sailor and command have both scored each other as a number one preference, then that match will be made.

Although rare, the NA can interject if a particular sailor and command that it has determined to be a good match have not interacted.  This particular function of the NA is done without human prodding.  The NA's human counterpart, however, closely controls most other functions.

## Conclusion

Creating a system of interacting intelligent agents of this magnitude is no simple endeavor. This paper has laid out the framework of how the system should work and how we plan on dealing with many of the major issues. As this project is still in progress, we do not know if our framework might need to be revised or even scrapped for a better one. At this stage, however, we have progressed far enough to be fairly confident with the general outline even if the details are still a bit fuzzy.

While the technical issues of a software project of this scale are daunting, we cannot be sucked into the trap of thinking that they are the only issues. The “system” described here does not consist of silicon and steel alone but also of blood and bone. As described in the previous section, the autonomy of the human agents must be considered along with the autonomy of our artificial agents.

Finally, it should be noted that there are no assurances of performance. Even though the system contains numerous safeguards for such, especially in the NA, it is still conjecture that this methodology will produce an acceptable result.

## References

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

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

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

Barsalou, L. W., 1999, Perceptual symbol systems. **Behavioral and Brain Sciences 22**:577–609.

Bogner, M., 1999, Realizing "consciousness" in software agents. Ph.D. Dissertation. University of Memphis.

Bogner, M., Ramamurthy, U. and Franklin, S., 2000, “Consciousness” and Conceptual Learning in a Socially Situated Agent. In *Human Cognition and Social Agent Technology*, edited by K. Dautenhahn (Amsterdam: John Benjamins).

Brainov S. and Sandholm T., 1999, Power, Dependence and Stability in Multiagent Plans. AAAI'99 (Orlando, FL), pp.11-16.

Castelfranchi C., 1995, Guarantees for Autonomy in Cognitive Agent Architecture. In *Intelligent Agents: ECAI-94 Workshop on Agents Theories, Architectures, and Languages*, edited by M. J. Wooldridge and N. R. Jennings (Berlin: Springer-Verlag), pp. 56-70.

Castelfranchi, C., 2000, Founding Agent's Autonomy on Dependence Theory. ECAI'01 (Berlin), pp. 353-357.

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

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

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

Franklin, S., Kelemen, A. and McCauley, L., 1998, IDA: A Cognitive Agent Architecture. IEEE Conf on Systems, Man and Cybernetics (New York, NY.: IEEE Press).

Hexmoor, H., 2000, A Cognitive Model of Situated Autonomy. PRICAI-2000 Workshop on Teams with Adjustable Autonomy (Australia).

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

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

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

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

Maes, P., 1989, How to do the right thing. **Connection Science, 1**, 291–323.

Mele, A., 1992, *Springs of Action* (New York: Oxford University Press).

Mele, A., 1995, *Autonomous Agents: From Self-Control to Autonomy* (Oxford University Press).

Minsky, M., 1985, *The Society of Mind* (New York: Simon and Schuster).

Mitchell, M., 1993, *Analogy-Making as Perception* (Cambridge, Mass.: The MIT Press).

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

Steels, L., 1995, When are robots intelligent autonomous agents*? Robotics and Autonomous Systems*, *Vol. 15*, (Amsterdam: Elsevier Science Publishers), pp. 3-9.