USER MODELING FOR MILITARY TRAINING: INTELLIGENT INTERFACE AGENTS

Sheila B. Banks, Martin R. Stytz, Eugene Santos, Jr., and Scott M. Brown
Artificial Intelligence Laboratory
Virtual Environments Laboratory
Department of Electrical and Computer Engineering
Air Force Institute of Technology
Wright-Patterson Air Force Base, OH 45433

INTRODUCTION

The current state of operation for users within a distributed virtual environment (DVE), especially those for the military training community, places an unmanageable cognitive burden upon the user. The user must attempt to understand the environment, which may contain thousands of both real world and synthetic components; extract relevant information from this complex environment; and, within a fixed time frame, analyze the information determined to be relevant. However, the user's purpose is typically to make decisions based upon the relevant information. Because the information is difficult to locate and has a short time period of relevance, human decision making necessarily suffers. While some advances in user interface design can alleviate some of this problem, the basic problem of information overload cannot be addressed simply through development of a better interface or with the use of ad hoc decision support tools.

Current intelligent user interface (IUI) research is primarily focused on human-computer interaction issues related to the abilities and usability of interfaces. Thus far, research to enable intelligent interfaces has minimally addressed the development of a basic knowledge structure suitable for representing the interface intelligence required to make the complexity of DVEs transparent to the user. Current intelligent interface agents lack the representational complexity to manage the uncertainty and dynamics involved in predicting user intent and modeling user behavior. Our work was undertaken to overcome these shortfalls.

An accurate cognitive model of the user is considered to be necessary for effective prediction of user intent. The problem with most cognitive models for intelligent user interfaces is that they rely upon rule-based intelligence structures. Rule-based representations for current intelligent user interfaces lack flexibility and power in two key areas: representation of uncertainty and dynamic user modeling. Employing a knowledge representation that correctly captures and models uncertainty in human-computer interaction can improve the modeling of the user and the user interface's behavior. One representation that is ideal for representing uncertainty is a Bayesian Network (BN). A BN is a mathematically sound portrayal of uncertainty that models the probabilistic relationships between items. Bayesian techniques have attractive properties for developing interface intelligence because they can capture uncertainty, which is required to model human intent. Also, Bayesian techniques are extremely useful in predicting future events. Finally, these techniques are useful for expressing qualitative relationships (causalities) among beliefs and for processing these relationships in a way that yields intuitively plausible conclusions.

To investigate and evaluate the use of BNs for intelligent user interfaces, we developed an intelligent agent called the Intelligent Interface Agent (IIA) that uses a model of user behavior to predict future user behavior or user intent and to suggest adaptations. An intelligent agent is a computer entity that collaborates with and helps a user. The roles of an intelligent agent include perception of dynamic conditions in the environment; action to affect the environment; and reasoning to interpret perceptions, solve problems, draw inferences, and determine actions. These intelligent agent roles support the needs of intelligent user interfaces. For the intelligent agent to fulfill its roles, it must have the ability to reason. The reasoning capability is enabled by the following activities: collecting domain metrics, transitioning metrics into a representation, storing information, and inferencing over the stored information. These actions work together to provide the DVE participant with an environment where the user interface can make intelligent decisions and enhance the decision making process of the user.

This paper will first provide the motivation and background for our current efforts and then present the knowledge representation for IIA that supports effective user intent prediction by incorporating the ability to model both the uncertainty in user intent and dynamic user behavior. The proof of concept of IIA has been implemented and the paper will also present the results of the implementation and the viability of the IIA. We also address the incorporation of this intelligent interface agent into modern user interface technology within a military training DVE. Finally, we will draw conclusions from the current IIA design and implementation and present promising avenues for future research in the development of an intelligent interface agent using Bayesian Networks.

BACKGROUND

It is widely agreed that basing decisions on an accurate cognitive model of the user is important for effective prediction of user intent and that the interface should be able to collect and model information about false inferences [15, 24]. Collecting such data is cognitively and computationally difficult. DeWitt notes that not all naturalistic (i.e., observable) properties play an interesting role in a user's causal model [5]. That is, only certain observable actions and information in a user's "world" will have relevance to that user. Therefore, to effectively and efficiently capture user intent, our model should not attempt to model every possible action the user may exhibit, but only those that are relevant, i.e., most likely to be exhibited.

Many research intelligent interfaces use rule-based intelligence [6]. Rule-based representations, like those used in most intelligent user interfaces, fail in two key areas: representing uncertainty and dynamic user modeling. The use of "probability modules" [25] is an ad hoc approach to determining answer reliability or uncertainty. Furthermore, the addition and deletion of rules to dynamically model a user is ad hoc. Therefore, knowledge representations that can dynamically capture and model uncertainty in human-computer interaction can improve the modeling of the user and user interface states in an intelligent user interface. One knowledge representation that is ideal for representing uncertainty is a Bayesian Network (BN)[1] [3, 16]. A Bayesian network is a mathematically correct and semantically sound model for representing uncertainty that provides a means to show probabilistic relationships between items.

An intelligent agent is a computer entity that collaborates with and helps a user by perceiving dynamic conditions in the environment; acting to affect conditions in the environment; reasoning to interpret perceptions and solve problems; drawing inferences; and determining actions [1, 10]. The roles of an agent match the roles of today's intelligent user interfaces: knowledge-based interaction, self-adaptation, and automatic generation [17]. Self adaptation involves the intelligent interface improving its own performance based on perceived and stored behavior. Automatic generation takes place when the interface uses its existing knowledge base to generate an interface that is particular to the current state of the knowledge base. An intelligent interface agent supports development of compact, portable intelligence that is independent of a particular application interface.

The PESKI environment is an intelligent system designed to develop intelligent systems [18]. PESKI is an integrated knowledge-based system framework that combines the functions of natural language interface, knowledge base inferencing, explanation and interpretation, and knowledge acquisition and maintenance into a single, consolidated application. PESKI integrates the following closely interrelated, yet specialized tools: (1) Intelligent User Interface, (2) Inference Engine, (3) Knowledge Acquisition, (4) Knowledge Base Verification and Validation, and (5) Data Mining. The PESKI environment was used for our initial tests concerning the implementation and usability of the Intelligent Interface Agent.

[1] See http://www.afit.af.mil/Schools/EN/AI/ for an excellent resource on Bayesian Networks.

INTELLIGENT INTERFACE AGENT REPRESENTATION

The overriding goal for our prototype intelligent user interface development was to assist the user with managing the complexities of a software system through the use of intelligence techniques. The intelligent user interface also provides access to the software system tools and applications within the prototype intelligent system environment.

The intelligent user interface is based upon an intelligent interface agent (IIA) architecture [8]. The IIA controls the communications and intelligence aspects of the interface and is composed of three layers: the adaptation layer, the adaptive layer, and the communications layer. The adaptation layer manages and tracks all adaptations the user makes to the IUI. The adaptive layer communicates directly between the IIA and the interface presentation to perform interface initiated adaptations to the IUI based on perceived user behavior. Finally, the communications layer controls the various modes of communication available to the interface such as structured text, graphical manipulation, and natural language.

IIA employs a knowledge representation and a domain metric protocol to manage intelligence to satisfy the intelligent interface requirements mentioned previously. The basic representation for the IIA's knowledge is a Bayesian-based network called the Interface Learning Network (ILN). Because user behavior is not deterministic, using an uncertainty-based network representation for user behavior is appropriate. This representation can portray a large amount of information using a small collection of interface domain metrics. The second element of the IIA's reasoning ability is a domain metric collection protocol. These metrics are called interface domain metrics. Interface domain metrics can be any type of data that a user interface can collect from the application domain or the user at runtime. The number and type of interface domain metrics collected is solely based on knowledge required for user interface reasoning. Information about the application domain can be acquired from a single interface domain metric or combinations of different types of metrics. The collected interface domain metrics then need to be transformed into meaningful information. In IIA, this transformation includes

updating the information maintained in the interface learning network. As a result of this continuous update of domain metric data, when the user interface agent must make a decision, the agent can draw upon the most recent knowledge stored in the IIA's network.

The IIA unique representation dynamically captures and models user behavior and dynamically captures and models uncertainty in the agent's reasoning process. IIA has the ability to alter its own topology to better adapt itself to modeling a particular user. IIA's sound semantics and mathematical basis enhance its ability to make correct, intelligent inferences about the user's needs.

The Interface Learning Network

The Interface Learning Network (ILN) is the heart of the IIA architecture. The Bayesian network knowledge representation captures, stores, and models user and interface behavior. The network is composed of two semantically different nodes: interface learning nodes and interface information nodes. The network is also composed of containers that store learned user and user class behavior data and a network communications facility.

Interface Learning Node. Semantically, the interface learning node represents behavior the interface has collected about a particular system user or class of users. This node is named according to the behavior collected. Each node's probability is stored as a fraction. The denominator of the fraction represents the number of learning occurrences that affect the node. The numerator of the fraction represents the number of learning occurrences that add to the truthfulness of the node (i.e., a higher probability).

After the node is instantiated, the interface learning network loads stored data about the current system user into the interface learning node. Whenever the system user exhibits behavior represented by the node, the interface will call the node's update method to record the behavior.

Interface Information Node. Semantically, the interface information node represents a possible user state. Each interface information node is supported by two or more interface learning nodes and zero or more interface information nodes. The node sits "dormant" until the interface queries it for its probability, in order to make inferences as to the user's future state, which may be interpreted as future user action or intent. When queried, the node combines the probabilities using Bayes Theorem. The resulting probability represents the probability the node's state is true. This node is named after the state it represents.

Dynamic Interface Learning Networks

The networks used currently [7, 8, 9] are pedagogical examples. While they represent the concepts and advantages of using BNs as a knowledge representation in intelligent user interfaces, they represent only a microcosm of the entire system and the possible actions a user may perform while using the system. A brute force method to creating an interface learning network to represent the user's actions is to have a single node for each action possible in the system. This ILN could potentially have thousands of interface information nodes and approximately as many interface learning nodes. This approach, creating a node for every possible action in our system and allowing this node's dependencies to be connected, provides exact representation and the most accurate representation of the possible actions a user could perform at any time. However, there are two main problems with this approach. First, since our user will rarely if ever exhibit certain actions, the probability of certain nodes will be very small. When combining probabilities, these "irrelevant" probabilities can have the effect of ignoring the relevant node's probability. In DeWitt's terms, these actions are not "causally efficacious." Secondly, it is well known that belief propagation in BNs is NP-hard [4]. Therefore, an approximation to our network that models only relevant nodes and is a good tradeoff between computational complexity and representation exactness is desired. This requirement necessitates a dynamic ILN structure, where we add and remove relevant nodes in our ILN. However, the methodology to determine what nodes are relevant can be difficult.

Methodology. As mentioned previously, there are computational limits of modeling every possible action the user may perform as an ILN. However, for any given user, that user will only display a subset of all possible actions during a given interaction with the system. Yet, the subset of possible actions may be too large to use as a basis of a complete ILN. Therefore, we must restrict our interface learning network further.

There has been much research in the field of approximating BNs [4, 19]. Current techniques revolve around stochastic simulation, Likelihood Weighting, and Logic Sampling. Since a user will only exhibit a subset of all possible actions, we only allow a total of the N most relevant nodes to be present at any time in our network. When a user performs an action, this action may or may not be represented in the ILN. If it is, we update the network and calculate the new probabilities. If it is not, we modify the existing ILN topology. In our current implementation, we limit the number of nodes allowed in the user's ILN at any time. If the user performs an action that is not represented in the current network, we add a node representing that action to the network. If we have reached our network size limitation, we delete the lowest probability node from the network. In this way, the most relevant and highest probability nodes are present in our network at all times.

Metrics. Using this methodology for dynamically changing a user's ILN, we now must find the best

method to represent a particular user and ensure the resulting ILN is truly representative of our user. We define several objective metrics (versus subjective usability comparisons addressed later in this paper) that give us insight into the performance of our ILN. Observation of these metrics give us insight into both the effectiveness and efficiency performance of our network.

Absolute thrashing is the result of an observable property, represented as a node, repeatedly entering and leaving the relevancy set. We are concerned with adding a node to the interface learning network only to have it never queried and leave the network some time later.

User thrashing occurs when a user's intent and the system's measure of intent, as represented by the interface learning network, swings from one extreme to another. We also desire to avoid thrashing of the system's measure of user intent so we can make accurate predictions of the user's intent. As a concrete example, consider an aunt who is known to have drastic mood swings. You are aware of this, and develop ways of observing her current behavior to find a promising way of approaching her. You never vary your approach drastically.

Rate of divergence is a measure of how quickly a node leaves the interface learning network. We are concerned with ensuring a node that was added to our network, but is not used often, will exit the network quickly and allow more useful nodes in the relevancy set.

Class thrashing is the result of a user who is diametrically opposed to the user class and, as a result, the network initially does not represent a user well. Consequently, the network must "learn back" a user's behavior. This type of thrashing not only affects the user by making incorrect inferences, but affects the user class where the misplaced user is currently a member.

Rate of convergence is how fast past observed behavior is overcome by changes in current behavior and how quickly the probability of a node will settle out to a particular value. This measure is important in conjunction with class thrashing. We desire to know how fast a network will allow a user to overcome past behavior. For example, a user may exhibit a particular behavior for a "long" time and then suddenly change behavior, perhaps as the result of some new stimuli in the user's environment. A fast rate of convergence will quickly allow the user model to overcome the past behavior and accurately model the current behavior.

USER MODELS AND USER INTENT

Determining how to construct a network to accurately model a user is difficult at best. Furthermore, once we have built a user model, how do we modify it to more accurately model a user? Two main schools of thought exist on the construction of user models. The first uses "hand-coded" user models. That is, the system designer determines how best to model the users by constructing the user models a priori. Examples of "hand-coded" user models include Jameson [11] and Maes [12]. Hand-coded user models are typically static. Once they are designed, they will not change structure. The second method uses "machine-coded" user models. That is, a user model is constructed by the system as it "learns" more about the user. These models are dynamic where the structure changes over time. Jameson [11] and Harrington [7] provide examples of dynamic user models. Both methods have advantages and disadvantages; however, we are concerned with issues that impact both hand and machine-coded user models. These issues are discussed below.

Relevancy and User Intent

As mentioned previously, to accurately predict user intent, we must have an accurate cognitive model of the user. Fortunately, modeling every possible naturalistic property in the user's world does not lead to the most accurate model [5]. If this were not the case, we would have little hope in using numerical uncertainty management techniques such as Bayesian Networks due to the computational inefficiency of large networks [18].

DeWitt users the term "causally efficacious" to describe naturalistic properties that play an interesting causal role in cognitive functions [5]. He argues that not everything observable is of interest when we make decisions. DeWitt's philosophical argument has direct analogy in user models. As mentioned previously, a user model should not include every possible piece of information about the user's world. To include information not causally efficacious to a user model needlessly complicates the model both semantically and computationally. In other words, the less causally efficacious a property is to another, the more likely it is we can ignore it.

We use the term *relevancy set* to describe those properties, represented as nodes, included in a user's interface learning network. Relevancy set and ILN are used interchangeably; however, the network fully captures the causality of the model (nodes, arcs, and probabilities), while the relevancy set only captures the nodes in the network. For hand-coded models, the relevancy set will not change. For machine-coded models, the relevancy set may change with use. A relevancy neighborhood are those properties that are immediately causally efficacious to the decision under consideration. As an example, a user's interface learning network may contain the possible communication modes and tools a user may use, given the user's class and individual preferences learned by the user's interface learning network, as well as several knowledge bases used previously.

The current implementation of our IIA uses a dynamic interface learning network that monitors "hand-coded"

user actions [2]. That is, we determine a priori the actions we will monitor to limit the number of user actions that must be evaluated in the system. IIA, while currently incorporating some "hand-coded" model features, is a dynamic structure that allows the dynamic addition and deletion of nodes as IIA learns more about the user. We limit the number of nodes allowed in the user's interface learning network at any one time. If the user performs an action that is not represented in the current relevancy set, we add it to the network. If we have reached our current network size limitation, we delete the lowest probability node from the network. This method ensures the most relevant actions are in our relevancy set at any given time.

User Classes

A user class is a generalization of a number of users sharing common characteristics. User classes have been used in a number of systems for various reasons [11, 12]. For example, the prototype implementation of IIA within PESKI segregates users into the following user classes: application users, application experts, knowledge engineers, and computer scientists [7]. User classes serve two main purposes. New users to the system will have their individual interface learning nodes set to the most uncertain probability. However, user class information learning nodes will contain probabilities of the class to which the user belongs. Therefore, the user class helps bias the agent initially towards this class. As the user works with the system, the preferences are captured in the individual user interface learning nodes and IIA makes better predictions concerning the user's intent. Analogously, a real-world personal assistant can not be expected to accurately predict an employer's actions the first day. However, based on previous employers with similar backgrounds, the personal assistant attempts to determine what information the employer will need and when. As the assistant learns more about the employer's behaviors, beliefs, and intentions, prediction becomes more accurate. Secondly, by collecting user class data, system designers can determine what communication modes, tools, etc. are being used by a particular class. This information can then be used systematically to motivate future improvements to the system.

Several problems can arise from user classification. First, a user may belong to more than one user class. For example, consider the computer scientist concerned with both the development of the intelligent system (requirements, design, implementation, maintenance) and the knowledge engineering of that system. As the user works with the system, some of the actions are best categorized by the computer scientist class, while others are best characterized by the knowledge engineer user class. Both the user and the user class may suffer as a result. Another major problem is the misclassification of users. IIA within PESKI allows the user to select the user class when using the system for the first time. If this user determines the user class incorrectly, it is possible that the actions may be diametrically opposed to those of the class in which the user has placed himself. Once again, both the user and the user class suffer. We desire to minimize these problems by identifying when they occur.

The Correction Model

This section addresses when and how to dynamically change the underlying reasoning mechanism(s) employed by IIA. We have developed several discriminators for determining when a particular problem is occurring in the network. Each discriminator is associated with a different metric presented previously (e.g., class thrashing, absolute thrashing) and has an associated method to correct the problem (e.g., changing the user's class, slowing the rate of divergence) and a utility value. This utility is a measure of the discriminator's "importance" with respect to the impact it has on correcting the learning network so it makes more correct suggestions.

The scenario for having a discriminator suggest changes to an incorrect interface leaning network is based on the concept of a contractual bidding process, where the discriminator with the highest utility "wins" the contract to correct the network. We model each discriminator as an agent. These agents engage in a "bidding process" to recommend changes to the interface learning network. A manager agent is responsible for determining when a contract is available, announcing the contract to be filled, receiving bids from the bidder agents, evaluating the utility of the bids, and finally accepting or rejecting the bids based on their utility.

A contract announcement is made when a correctness metric is below an empirically chosen threshold. This metric is an exponentially smoothed average of the number or correct suggestions to the total number of suggestions made by the interface agent. Bids are made by the bidder agents. When the manager agent has received all bids or the contract announcement has expired, the manager determines the bidder agent with the best bid, represented by the highest utility, and awards the contract by allowing this bidding agent to effect its changes to the network.

The utility is based on the correctness metric's magnitude of improvement if the changes were implemented at some point in time prior to the correctness metric falling below the threshold and the agents' past effectiveness in providing changes to the network. The manager evaluates the utility by making the proposed changes to the "oldest" interface learning network stored in the history. Then, for each suggestion made to the user, the manager determines what the new suggestion(s) would be, based on the bidder agent's proposed changes and any evidence

stored in the history. The utility is then simply the number of correct suggestions to the number of suggestions made over the history. This utility is multiplied by the bidder's effectiveness. The agent's effectiveness is updated by simple reinforcement learning, where the "winning agent" receives positive learning. The reinforcement learning takes into account those bidder agents that are determined to be "helpful."

We can adopt any number of bid strategies. Currently, the negotiation protocol function and negotiation strategy are defined generally the same as Muller [13], which is a sealed-bid, single award strategy. That is, the other agent have no idea what "price" (utility) the other agents are bidding.

USABILITY STUDY OF IIA

Use of the IIA does not necessarily ensure a usable interface for a software system. Therefore, usability testing is required. Usability has been shown to be a prime factor in determining the user acceptance of interaction devices and the form of interaction [14]. In traditional usability studies, a series of interface prototypes coupled with user evaluation are usually necessary to determine the final design. Our work pioneers the application of usability concepts and metrics to the design and evaluation of an intelligent interface agent rather than to merely the interface. To simplify initial testing procedures, these IIA usability studies were conducted within the PESKI environment.

Usability testing

IIA usability testing was performed with standard time/step analysis and user feedback sessions. There were four tests used to evaluate the usability of the interface intelligence. The first test quantifies procedures the user must follow to get work done. User acceptance of the interface intelligence is captured in the second test. The third test measures the responsiveness burden the intelligence places on the interface. The final test examines how closely the model actually represents user intent. Harrington, et. al. [9] gives detailed results concerning the usability study and includes the following information: study sample size, data collected for physical work requirements, surveys and data collected for acceptance and responsive testing, and measurements taken from the interface intelligent network for accuracy testing.

Physical work requirements and results. Collecting data about the physical work a user is required to do is one way to evaluate the usefulness of the interface intelligence. Physical work requirements such as keystrokes, menu selections, reading, and button presses were collected for a user operating the interface intelligence. The current implementation of the IIA makes suggestions pertaining to what system function, communication mode, and file the user wants to access at system startup. Our results show that using the IIA's suggestions yields a considerable savings in physical work for the user.

Acceptance testing of the IIA and results. User acceptance data was collected by exposing a number of users to the interface intelligence and eliciting user opinion on a written survey. We conducted a pilot user acceptance study for the IIA. The test results are divided into three general areas: timeliness of operations, complexity of operations, and usefulness of the IIA.

Users were generally satisfied with the timeliness of operations, although they seem to find the automatic operation perform by the IIA slightly slower than manned performance of the same operations. Users generally found the IIA to be useful, although these results are most probably influenced by the results for user opinion on timeliness and complexity.

Responsiveness testing of the IIA and results. Responsiveness of an interface is an important criteria for interface users because it indicates how well the interface helps the user accomplish tasks. Our testing of the responsiveness of the interface was based on user opinions and empirical data. The user opinions are collected in a manner similar to that for user acceptance data and empirical data is taken by collecting real time data during interface functions that are influenced by the interface intelligent network. Together, these data provide a good indication of the acceptability of the intelligent user interface's responsiveness.

The results of the responsiveness study for the IIA indicate that IIA creates some user noticeable pauses. The pauses arise from update calls to the inferencing mechanism and execution calls for the graphical communication mechanism. The user acceptance study shows that users found the pauses noticeable but acceptable.

Accuracy testing of the user model and results. A study of the IIA's ability to predict future user behavior is necessary to determine if the IIA accurately models user intent. This can be accomplished by observing the dynamics of the agent's suggestion generation capabilities when given a set of test cases that mimic user behavior. Two representative test cases used for this research to explore the accuracy of the user model were *single focus* and *double suggestion*.

The first test evaluates IIA's ability to rapidly adapt to a user's change in preferences between two options. In the single focus case, the probabilities of two interface network nodes are tracked through the case of a user change of preference. The user begins by selecting the Data Mining function and, after the initial 15 times, the user switches preference to the Inference Engine. For this case, only the function suggestion is evaluated. The results of this test are shown in Figure 1. These results show the IIA's ability to adapt to the user's change in preference. In this case, the IIA is able to

respond in only two iterations to the change in user behavior.

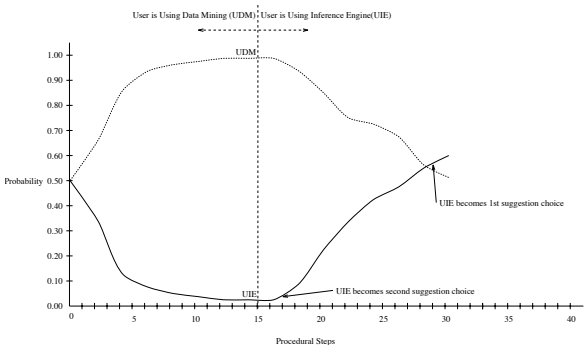

Figure 1: Overcoming Evidence to Adapt Single Focus Case.

The double suggestion test case differs from the previous in that the focus of the user is on a combination of suggestions that may include filename, system function, and communication mode. A combination or *double suggestion* must be completely true for the user to accept it. In this case, the user's first 10 selections are for the Inference Engine function with Text Communication and filename "Full5.bkb" and then switches preference to the Knowledge Acquisition function with Graphical Communication and filename "Afit.bkb." The results of this test are shown in Figure 2. These results show the IIA's ability to adapt to the user's change in preferences. Also, the acceptance and rejection of suggestions, especially the rejection of suggestions that are partially but not fully correct, affectthe probability distribution throughout the network.

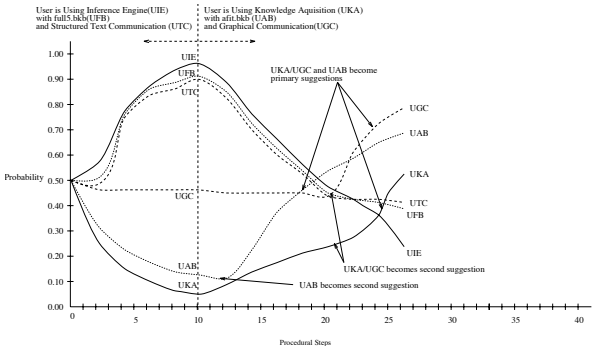

Figure 2: Overcoming Evidence to Adapt Double Suggestion Case.

Usability Results

The usability testing performed for IIA indicates great promise for this interface intelligence agent. The mathematical accuracy and ability of the IIA to model user intent show the appropriateness of applying Bayesian techniques to intelligent user interfaces. The responsiveness, work, and user preferencestudies show that the technique is acceptable, but requires refinement in responsiveness. Finally, the ability to capture uncertainty through Bayesian techniques has immense value to the accuracy and dynamics of IIA's predictions. The usability testing also shows that the intelligent user interface can adapt its suggestion based on changes in the user's behavior. This supports the claim that the IIA does model user intent. However, the test results also show the changes occur only gradually. Additional accuracy testing also highlighted that misuse of the IIA can lead to an inaccurate model.

INCORPORATION OF IIA INTO A MILITARY TRAINING DVE: SIRDS

An ideal DVE interface for the military training environment would enable the user to perform a wide variety of useful work without hindering the observation of the virtual environment. The interface would provide convenient access to virtual environment display parameters, analysis outputs, conferencing and collaboration capabilities, intelligent agents, motion and orientation controls, and situation awareness aids. The interface would also allow the user to directly manipulate objects within the virtual environment and provide familiar interface mechanisms.

To achieve these objectives, we require a comprehensive software engineering, knowledge engineering, and knowledge acquisition methodology for Symbiotic Information Reasoning and Decision Support (SIRDS). The name for the methodology is descriptive of our intent. The interface should be symbiotic, that is work tasks should be appropriately partitioned between the computer and the user. The computer requires the user to provide guidance and insight into the information that is necessary and to draw complex, or high level, inferences from the data. The user, on the other hand, looks to the computer to perform data acquisition and management, low level quantitative and qualitative data analysis, and routine inference to enable decision support, as well as to manage the data and its display. A symbiotic approach is necessary because the objective is to let the user and the computer share the taskload. The key interface design question is identifying a human-centered task partitioning between the computer and the human. The Information Reasoning component of the interface deals with the issues related to abstracting and analyzing information. The Decision Support aspect relates to the need to enable the user to understand the relevant data and to perform necessary analysis to allow the system to provide information highlighting and user focus of attention support.

Achievement of SIRDS requires the development of an adaptive, intelligent, learning man-machine interface. SIRDS addresses a wide range of cognitive issues, as well as data fusion, ambiguity resolution, representational mapping, mixed initiative dialogue, and other agent and data visualization factors that must

be considered. Construction of the interface requires a mix of traditional human-computer interaction, data visualization, and intelligent agent capabilities within a software engineering framework. The framework supports the symbiosis of human cognition and computational power required to deal with complex DVEs. Intelligent agents are a key aspect of SIRDS, and they perform information fusion, analysis, and abstraction, as well as deriving information requirements and controlling information display. Agents within SIRDS are of two types, one for the task of reasoning to direct system data acquisition, assessment, and information synthesis; and the other for reasoning about information display. However, the same software architecture and development methodology is employed to realize both types of agents.

In addition to performing information retrieval and analysis, SIRDS relies upon information visualization techniques to enable the user to understand the derived information and available processing options. To maximize user effectiveness in an information dense environment, SIRDS also operates in anticipation of user information needs. To do so, SIRDS must first ascertain user information requirements, initiate data retrieval operations, and assist with analysis of the resulting relevant information.

The first step toward realizing this vision is the development of a design methodology for intelligent agents for the control of information display and for providing user assistance in the integration of and access to information. In our view, the development of agents for information display should be performed in parallel with the design and development of the traditional aspects of human-computer interaction and information presentation. To be comprehensive, the methodology must address information representation and visualization by the interface agents, the software design of these agents, usability criteria for agents, and metrics for determining the necessary agents for interface control.

We have begun to address these needs by developing various prototype systems within our Labs. The Information Pod [20] addresses the need for a user interface software architecture and the Sentinel [21] the need for information integration and analysis. The Intelligent Interface Agent (IIA) addresses the need for determining user intent and intelligent information presentation. IIA is integrated with the Information Pod and Sentinel projects to produce an intelligent user interface to a DVE. The Synthetic BattleBridge (SBB) [22] virtual environment project, which uses DVE technology [23, IEEE 1278-1993] to achieve a complicated, purposeful virtual environment, is the host military training DVE application for SIRDS. We believe that our approach is scaleable in breadth and in depth across the required elements of software architecture; intelligent information integration and analysis; and intelligent information presentation and user modeling.

CONCLUSIONS AND FUTURE WORK

In this paper we described the knowledge representation necessary for user modeling and for the prediction of user intent to create an adaptive distributed virtual environment user interface. This intelligent agent, IIA, provides an effective knowledge representation for user, user class, and interface behavior. The use of Bayesian networks over rule-based systems to accurately model the user better captures the uncertainty of user actions by using sound semantics and a firm mathematical basis. Initial tests show noticeable savings in the user's physical workload while accurately predicting users' behavior. We have presented several metrics that provide an insight into the performance of our network. Furthermore, the momentum of learned behavior in one direction can be reversed and changed to another direction of behavior quickly.

The research undertaken arises from the vision that DVEs provide a potentially revolutionary means for humans to interact with each other and with computers within a military training environment. However, to achieve this potential, techniques that allow users to accomplish a wide variety of work and communication within a DVE must be developed. Our approach to an adaptive user interface for virtual environments, SIRDS, provides one means to allow users effective access and use of the virtual environment and its data. Prototypes within SIRDS development have shown capabilities in a wide range of issues that includes user intent inferencing and user model. These capabilities form the basis of a self-consistent, adaptive, learning interface system for virtual environments.

Future efforts propose a dynamic meta-level of inferencing, capable of modifying the user's ILN topology as the user performs actions. To realize this, we must be able to determine "real-time" what is happening with the user. Most usability studies are done "off-line" and have no immediate bearing on the user model. In addition, the ability of this network to model user behavior can be expanded by designing the interface to understand the user behavior. For example, if the interface measures patterns of indicator swings the interface may begin to classify these patterns. The interface may then be able to assign patterns to user traits, such as moods. The incorporation of temporal reasoning into this representation would allow the interface to predict user traits based on the patterns [26].

REFERENCES

1. Baecker, R. M. and others. *Readings in Human-Computer Interaction: Toward the Year 2000, Second Edition*, San Francisco, CA: Morgan Kaufmann Publishers, Inc., 1995.

2. Brown, S.M., Harrington R.A., Santos, E. Jr., and Banks, S.B. "A dynamic Bayesian intelligent agent," *Proc. of Interfaces '97 - Man-Machine Interaction*, Montpellier, France, May 1997, pp. 118-121.
3. Charniak, Eugene. "Bayesian Networks without Tears," *AI Magazine*, *Winter*:50-63, 1991.
4. Cooper, G.F. "The computational complexity of probabilistic inference using Bayesian belief networks," *Artificial Intelligence*, 42:393-405, 1990.
5. DeWitt, R. "Vagueness, semantics, and the language of thought." *Psyche*, 1993.
6. Gonzalez, A.J. and Dankel, D.D. *The Engineering of Knowledge-Based Systems*, Englewood Cliffs, NJ: Prentice Hall, 1993.
7. Harrington R. A., Banks, S. B. and Santos, Jr., E. "Development of an Intelligent User Interface for a Generic Expert System," In M. Gasser (Ed.), *On-line Proc. of the '96 MAICS Conference.* URL: http://www.cs.indiana.edu/event/maics96/Proceedings/ harrington.html, 1996.
8. Harrington, R.A., Banks, S.B., Santos, E. Jr. "GESIA: Uncertainty-Based Reasoning for a Generic Expert System Intelligent User Interface," *Proc. of the 8th International Conference on Tools with Artificial Intelligence*, Toulouse, France, 16 - 19 November, 1996, pp. 52-55..
9. Harrington R. A., et al. *The PESKI Intelligent User Interface.* Technical Report AFIT/EN/TR96-03, Department of Electrical and Computer Engineering, Air Force Institute of Technology, 1996.
10. Hayes-Roth, B. "An architecture for adaptive intelligent systems," *Artificial Intelligence*, *72*:329-365, 1995.
11. Jameson, A. "Numeric uncertainty management in user and student modeling: an overview of systems and issues," *User Modeling and User-Adapted Interactions*, 5, 1995.
12. Maes, P. "Agents that reduce work and information overload," *Communications of the ACM*, 37(7):811-821, 1994.
13. Muller, J.P. "A cooperation model for autonomous agents." In J.P. Muller, M.J. Woolridge, and N.R. Jennings (Eds.), Intelligent Agents III: Agent Theories, Architectures, and Languages, *Proceedings of the European Conference of Artificial Intelligence '96 Workshop*, August, 1996, pp. 245-260.
14. Nielsen, J. *Usability Engineering*, Cambridge, MA: Academic Press Professional, 1993.
15. Opperman, R. "Adaptively supported adaptivity," *International Journal of Human-Computer Studies*, 40:455-472, 1994.
16. Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, San Mateo, CA: Morgan Kaufmann, 1988.
17. Puerta, A.R. "The Study of Models of Intelligent Interfaces," In *Proceedings of the 1993 International Workshop on Intelligent User Interfaces*, 1993.
18. Santos, E. Jr., and Santos, E. Sr. *Bayesian Knowledge-Bases*, Technical Report AFIT/EN/TR96-05, Department of Electrical and Computer Engineering, Air Force Institute of Technology, 1996.
19. Santos, E. Jr., Shimony, S.E., and Williams, E. "Hybrid Algorithms for approximate belief updating in Bayes nets," *International Journal of Approximate Reasoning*, to appear, 1997.
20. Stytz, M.R.; Banks, S.B.; Kesterman, J.J.; Rohrer, J.J.; and Vanderburgh, J.C. "Requirements, Design, and Implementation of the Information Pod Interface," *Proc. of the 7th International Conference on Human-Computer Interaction*, San Francisco, CA, 24 - 29 August 1997, to appear.
21. Stytz, M.R. and Block, E. "Providing situation awareness assistance to users of large-scale, dynamic, complex virtual environments," *Presence: Teleoperators and Virtual Environments*, 2(4):297-313, Fall 1993.
22. Stytz, M.R., Block, E., Soltz, B., and Wilson, K. "The synthetic battlebridge: a tool for large-scale virtual environments," *IEEE Computer Graphics and Applications*, 16(1):16-26, January 1996.
23. Stytz, M.R. "Distributed virtual environments," *IEEE Computer Graphics and Applications*, 16(3):19-31, May 1996.
24. Thomas, C.G. "Design, implementation, and evaluation of an adaptive user interface," *Knowledge-Based Systems*, 6:230-238, 1993.
25. Winston, P.H. *Artificial Intelligence, 3rd Ed*, Reading, MA: Addison-Wesley, 1992.
26. Young, J.D. and Santos, E. Jr. "Introduction to temporal Bayesian networks," In M. Gasser (Ed.), *On-line Proc. of the '96 MAICS Conference.* URL: http://www.cs.indiana.edu/event/maics96/ Proceedings/Young/maics-96.html, 1996.