

# JOINTADVISOR AN INTELLIGENCE ANALYSIS AGENT

**Robin R. Penner**  
**Erik S. Steinmetz**

Iterativity, Inc.  
Minneapolis, MN

## ABSTRACT\*

*This report describes the development of a multi-agent system, JointAdvisor, which provides decision support and integration for military dynamic decision-making. The intelligence cycle tasks that are supported include requirements analysis, collection planning, and intelligence analysis. It is a fully functional first iteration, including the ability to flexibly define probabilistic reasoning structures that incorporate Bayesian calculations. In addition to its realization of theoretical work in previous years of this effort, JointAdvisor extends the state of the art in agent architectures by providing semantic situational elements that manage and compute their own Bayesian probabilistic reasoning.*

## INTRODUCTION

An important research result of the Army Federated Laboratories Advanced Displays Consortium between 1996 and 2000 was the investigation of sophisticated decision support capabilities for intelligence analysis. These capabilities were prototyped in numerous demonstrations, including CoRaven (Hayes et al., 2000) and FOX (Fiebig et al., 1999). We were particularly interested in CoRaven, which prototyped the capability to build a reasoning structure using a commercial Bayesian reasoner (Hugin).

CoRaven showed that the display of Bayesian reasoner results on a map would be beneficial to intelligence analysis. CoRaven's implementation was limited to one simulated scenario and a set of hard coded message inputs, although the researchers involved in this investigation had performed a number of task and user-system allocation

analyses. These efforts provided the raw data to support the implementation of an integrated decision support tool for intelligence analysis.

We conducted several integration experiments to determine whether the results of the CoRaven program would scale up to actual conditions. If so, we were interested in whether an integrated tool could be implemented that leveraged these research findings. A secondary goal was the development of a solid platform for further research in human decision augmentation, multi-modal distributed collaboration, and automated decision enhancement through displays and advanced reasoning algorithms. A third goal was the development of an integrated Bayesian reasoning tool, which would allow us to experiment further with application of probability algorithms to situation prediction. Finally, we were interested in further exploring advanced concepts of multi-agent communication, including the use of shared generative situation representations.

The result of this six-month program of applied research is the JointAdvisor software (Joint Automated Decision Visualization: Intelligence, Synchronization, Operations, and Reasoning). JointAdvisor is the first iteration of an integrated collaborative tool for commanders, intelligence analysts, and collection managers at the Joint Forces Brigade level. It allows users to define Priority Information Requests (PIRs), to break these PIRs down into solutions, indicators, and observables, and to generate collection asset orders (SORs). When messages associated with these SORs are received, JointAdvisor presents the unfolding situation in multiple contexts to facilitate human reasoning.

The JointAdvisor program followed SEI Level 4 development processes, facilitated by adaptation of the Unified Modeling Process (Jacobson et al., 1999) and use of the TogetherJ design tool. The software is written entirely in Java, and so runs on any modern hardware platform. Although it is a first iteration, JointAdvisor is fully functional and could be immediately integrated into a larger environment (such as ASAS).

---

\* Prepared through collaborative participation in the Advanced Displays and Interactive Displays Consortium sponsored by the U.S. Army Research Laboratory under Cooperative Agreement DAAL01-96-2-0003. This work was performed at the University of Minnesota, Department of Mechanical and Industrial Engineering.

## JOINTADVISOR DESIGN

This first iteration of JointAdvisor began with a functional requirements analysis, resulting from the study of available domain documentation on the intelligence analysis process, interviews with intelligence analysts, and study of the original CoRaven scenarios. The high level functional requirements for JointAdvisor included:

- Provide an *information integration and decision support tool for Intelligence Analysis functions* to support the commander and S2 user.
- Provide mechanisms to *enhance the ability of intelligence analysts to think about, visualize, and analyze the situational picture.*
- Provide *integrated and appropriate views of the current and predicted situation*, including friendly and enemy objects, terrain features, collection information, messages and evidence, conclusions, predictions, and other situational elements.
- Provide mechanisms to compute and display *Bayesian probability information* and other automated reasoning contributed by decision support systems.

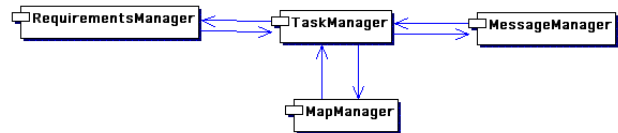
These requirements were extended and extensively reviewed with the development team and subject matter experts.

After the functional requirements were determined, we began the initial six-month iterative process of multiple cycles of use case development, activity and scenario diagram development, class definition, operational development, and evaluation. In accordance with the Unified Software Development Process (Jacobson, Booch, and Rumbaugh, 1999), all components and object interactions were iteratively diagrammed as sequence and activity diagrams. The use of such diagrams makes the system and component design explicit, allowing it to be easily understood and easily modified in subsequent iterations.

Architecturally, JointAdvisor is an object-oriented collaboration of multiple software agents. The agents in JointAdvisor, and their relationships with other agents, are shown in Figure 1. The currently implemented task agents in JointAdvisor include a visualization agent (MapManager), for spatial information; a requirements agent (RequirementsManager), for management of PIRs and their assertion structures; and a message agent (MessageManager), for input of evidence into the situation as it is received.

The task agents take autonomous action, communicating with situation objects through a central Task Manager Agent. JointAdvisor also provides user interface agents for

each processing agent, to separate the tasks of formulating communication and presenting interactions.



**Figure 1. JointAdvisor Agents**

We employed a central semantic repository and an active situation model (Penner & Steinmetz, 2000), to assure agent synchronization by providing a grounded basis for agent activities. In this architecture, types of domain objects are stored in a domain model, and then instantiated into an active situation model when needed. The JointAdvisor domain model is shown in Figure 2.

The JointAdvisor domain model contains definitions of the types of objects that can appear in the intelligence analysis situation. For example, at the top left, in blue, general military situations are defined, including operations, missions, and phases. The Phase class is defined as “having” several types of Information classes, including a number of PIRs and a TerrainMap (shown by diamond-tipped arrows going from Phase to the individual information classes).

Other information types in JointAdvisor include Messages, SORs, and Probabilities, all shown in pink at the top left of the diagram. Physical objects that can currently appear in the situation (green, lower left) include ObservableObjects, Observers, and NAIs (Named Area of Interest, a particular location on a map). Finally, Assertion situation classes include Solutions, Indicators, and SIRs.

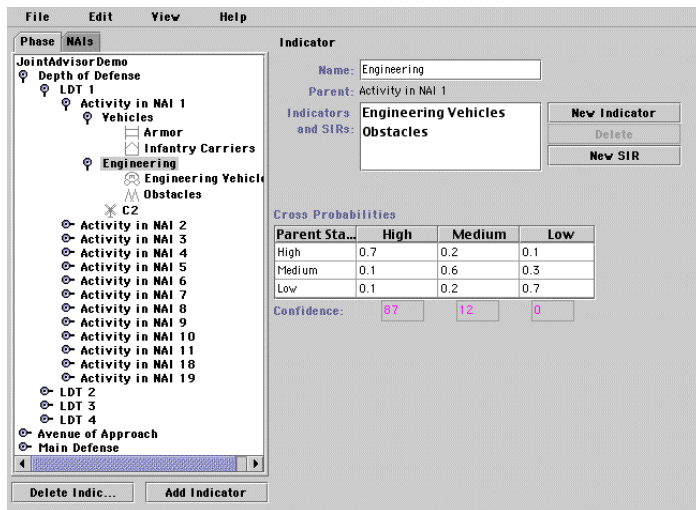
The objects in the domain model are used by the various task agents to build the dynamic situation model, and, through the Task Agent, to change objects in the currently active situation. For example, when a Message is received, a new Message object is created and added into the situation model, based on the generic Message class in the domain model.

Another example is when a user, through the Requirements Manager, changes a probability associated with an assertion (e.g., a Solution). In this case, the Requirements Manager notifies the Task Manager, which changes the properties of the Probability object in the situation. Because of internal operations in the situation, other objects will also change because of this probability change, and these changes will be communicated to the Requirements Manager, through the Task Manager, and shown to the user through the Requirements Manager’s presentation agent.



Note that, in many cases, the effort in this first iterative implementation of JointAdvisor concerned functionality and did not explore optimized user interface presentation and interaction mechanisms. This is particularly true for the entry of cross probabilities and visual coding of assertion state. Because a review of the literature did not reveal any previous approaches to the entry of Bayesian probabilities by human users, a strictly text based implementation was chosen. Our plan is to perform experimental usability research to determine appropriate input mechanisms for this type of information.

The definition display for an Indicator, the parent assertion for SIRs, is shown in Figure 4.

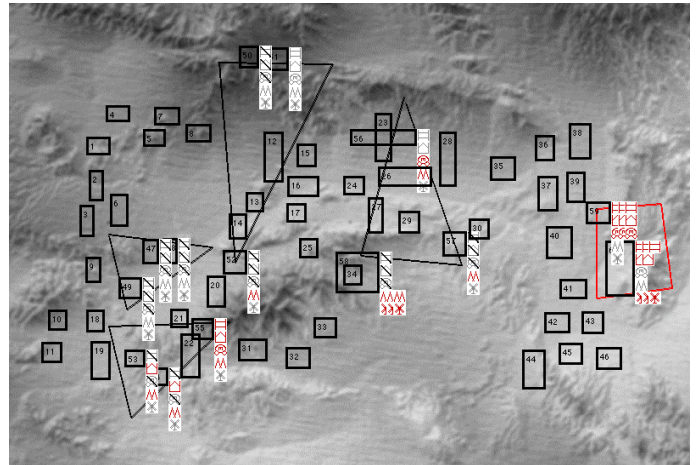


**Figure 4. Requirements Display for an Indicator**

In this example, incoming evidence has caused the elements in this assertion tree to recompute their probabilities. These new values are visually coded to indicate to the user that these are the result of system calculations. The calculated results are based on Bayesian probability algorithms, and show what probability the system believes that each possible state of the Indicator or SIR is true. Cross Probabilities, which are user changeable, are used to define the effect a lower level node has on higher level calculations of the best solution to the PIR. In JointAdvisor, each probabilistic object computes its own states and confidences; this is discussed further below. In the example above, the evidence involved Obstacles in NAI 1, and the Indicator that relies on this type of evidence (Engineering activity in NAI 1) has been computed to be of High probability, with an 87% confidence.

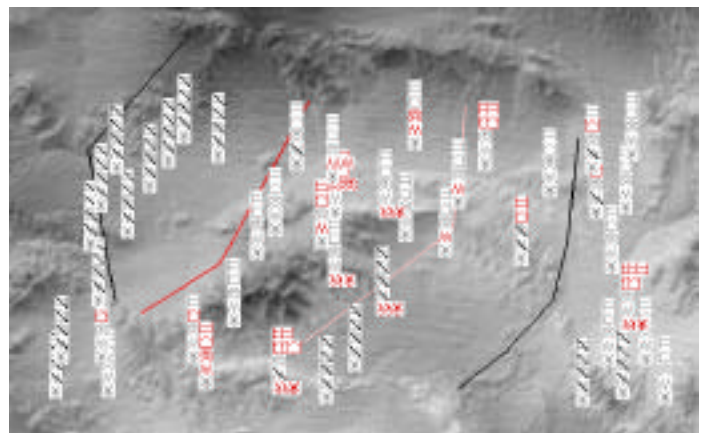
JointAdvisor presents information in multiple ways to facilitate human information processing and to increase the benefits of the automated reasoner. For example, it both graphically and textually displays the observed data, using design rules appropriate to each type of information. In addition, it also shows the results of Bayesian analyses,

again in both graphic and textual formats, to facilitate use of the information and integration into the full situational picture. Figure 5 is an example of a display generated by JointAdvisor’s Map Manager agent, showing the terrain and NAI locations, the requested observations/SIRs (gray icons), observations (red icons), and the state of the linked solutions to the PIR “Where is the main defense?” Solutions are shown as polygons, coded as gray (low probability of truth) to dark red (high probability of truth). This same information is available through the Requirements Manager, since each agent uses the same situation model, but each agent adapts the presentation to the task they manage.



**Figure 5. JointAdvisor Spatial Display**

Figure 6 shows another example of a spatial display, showing the situation associated with the PIR “What is the depth of the main defense?” This example highlights the ability of the spatial display to provide user-selectable information filtering; in this example, the user has selected to see only the evidence and conclusions, and not the NAI involved in this question.



**Figure 6. Spatial Display Without NAIs**

As with the user interaction selected for modification of Bayesian prior probabilities, the user interface elements

that are appropriate for indicating visual assertion state have not been fully explored. Because of a lack of literature on optimization of this type of display, a default coding scheme was selected for this iteration. Rather than a chromatic code as had been used in previous prototypes of this functionality, we selected a size code to indicate observable object state. This code is still not optimal for this information, but is an improvement on the use of color for identity. Further research is required to fine tune this and many other aspects of the JointAdvisor interface.

## JOINTADVISOR BAYESIAN REASONING

Although the earlier proof of concept system CoRaven depended on external software (Hugin) for reasoning capabilities, JointAdvisor incorporates Bayesian reasoning into any object that is probabilistic. The main types of probabilistic elements (Solutions, Indicators, and Observables) are subclasses of an Assertion object, which contains knowledge about Bayesian calculations for that object. In this way, we can both easily change the reasoning algorithm for experimentation, and encapsulate needed functionality so that it can be available to a number of objects.

The nodes in a JointAdvisor assertion tree, starting at a PIR on down, represent nodes in a Bayesian belief network. Each node is assigned a number of states that represents the different possible values for that node. Each of these states holds a probability that represents the belief that the node is in that state. For example, a solution may have the states “true” and “false”, and each of these states will have a probability reflecting the confidence that the solution is in that state. Each node also holds an array of values which represent the cross probability of the various states of the node. A cross probability represents the probability that, given that the parent node has some value  $x$ , what is probability of this node being in state  $y$  (having value  $y$ ). This is often written in the literature as  $P(y|x)$ .

In order to properly calculate the belief values for a node, it is also required to keep track of two more values for each state of the node: a value representing the effect of the tree above our parent node, and a value representing the effect of our children nodes and their descendents. These are often referred to as the  $\pi$  and  $\lambda$  values respectively. The belief value of a node is the  $\pi$  and  $\lambda$  values multiplied together for any given state, and then normalized so that the total of the belief values in a node sums to one.

Belief values of the nodes in JointAdvisor are updated using the message passing algorithm outlined in Pearl (1988). When a message arrives at a leaf node, we pass ‘ $\pi$ ’ messages up the hierarchy which trigger ‘ $\lambda$ ’ messages to be sent back down the hierarchy to children who have not already received or produced a message. In

this fashion, the belief values of the entire tree are updated in one pass with each new piece of evidence. JointAdvisor allows the user the option of specifying whether the Solutions of a PIR or the PIR itself will be considered the root of a Bayesian hierarchy. If the PIR is considered the root of the hierarchy, the Solutions are considered ‘linked’: the sum of the belief values of the solutions will be one. However, in those cases where the intelligence officer considers the possible solutions to a PIR to be independent, the solutions can be ‘unlinked’, allowing more than one solution to be considered highly probable or improbable.

## NEXT STEPS

Work on the JointAdvisor software should continue the development process that has been successfully applied to this iteration, providing iterative extensions and improvements, including:

- **Bayesian Reasoner Improvement.** The architecture of the JointAdvisor system allows sophisticated integration of probabilistic reasoning decision support. The currently implemented algorithms have been thoroughly tested from a mathematical standpoint, but not from a user or task standpoint. While predictive probabilities appear to help decision makers, it is unclear how to maximize this benefit and minimize negative effects of machine aiding. Improvements to the situational specificity of probabilistic computations are required; for example, the prior probabilities of a tank appearing in a defensive situation are different for the same type of tank in a different situation. Extensive investigations into probability values for default objects and situations are also required, including the automation of human entry of probabilities and the reverse engineering of probabilities from historical data.
- **User Interface Improvement.** While the user interface to the current JointAdvisor system has been fully “usability engineered”, emphasis on the interface was secondary to functionality in this first iteration. Much work remains on user interface mechanisms to modify probability distributions (e.g., the prior probabilities required for each Bayesian object) and to display the results of automated reasoners and decision support systems. For example, display of the state of a SIR on the JointAdvisor map display is not optimal, and visual coding for solution state has not been optimized to support user reasoning.
- **Integration and Extension.** JointAdvisor is currently a stand alone Java application. It can read and create XML, and so could easily be integrated with other tools such as FOX or ASAS. It has a rudimentary message agent that could be modified to understand and create USMTF, and requires extensions to its

Requirements and Analysis agents to allow them to provide reports and summaries to other military applications. In addition, major extensions to this system are required to make it applicable to real situations; for example, this iteration only contains a few observable objects and observers, and is limited to one geographic area.

We believe that this first iteration of JointAdvisor demonstrates its feasibility and usefulness, and that further development work on this system, if possible, will result in truly revolutionary capabilities.

## REFERENCES

Fiebig, C., Hayes, C., and Winkler, R. (1999). What's New in Fox-GA? In *Advanced Displays and Interactive Displays Consortium: Proceedings of the 3<sup>rd</sup> Annual FedLabs Symposium*.

Jacobson, I., Booch, G., and Rumbaugh, J. (1999). *The Unified Software Development Process*. Reading, MA: Addison-Wesley Publishers.

Hayes, C.C., Penner, R., Tu, N., Ergan, H., Lu, L., Asaro, P., and Jones, P. (2000). Model based design of decision support for real time information assessment. Submitted to *International Journal of Systems, Man, and Cybernetics*.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco: Morgan Kaufmann Publishers.

Penner, R. and Steinmetz, E. (2000). Dynamic user interface adaptation based on operator role and task modeling. *Proceedings of Systems, Man, and Cybernetics*, Nashville, TN, October.

---

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U. S. Government.