

Executable Models of Influence Nets Using Design/CPN

Lee W. Wagenhals, Insub Shin, Alexander H. Levis

System Architectures Laboratory
School of Information Technology and Engineering
George Mason University
Fairfax, VA 22030, USA

Abstract

This paper describes a methodology for converting an influence net to an executable model, implemented using the Colored Petri Net formalism and tools, so that it can be used to assess the impact of a set of controllable events or actions on outcomes of interest; specifically the impact of various sequences and timing of those actionable events. With this methodology, alternative courses of action, first defined using influence nets, can be refined by adding sequence and timing information for analysis and comparison. The techniques developed offer a means to integrate intelligence and operational planning models to improve course of action development. The paper includes a description of the automated algorithms that convert an influence net to a Colored Petri Net and illustrates how that model can be used for the analysis of alternative courses of action.

1 Introduction

In the command and control environment, planning and selecting specific courses of action to achieve objectives and goals involves two types of modeling and analysis activities (Figure 1). The first involves models that attempt to assess potential events and outcomes based on incomplete and uncertain understanding of both physics based and perception based processes. Such activity is primarily the forte of intelligence analysts. Probabilistic modeling techniques that provide inferences are often used to suggest what outcomes might occur given sets of controllable actions and uncertain exogenous events. The second type of activity, planning, involves the generation and evaluation of detailed actions and activities to accomplish the mission goals. In general detailed models and algorithms are available for planning the courses of action to be taken.

These two classes of activity involve different ways of viewing problems, and the models created have different characteristics. In many cases, the executable models used in planning require parameters that can be derived from the probabilistic models. It follows, then, that being able to place both modeling techniques in a common environment can enhance the overall planning process.

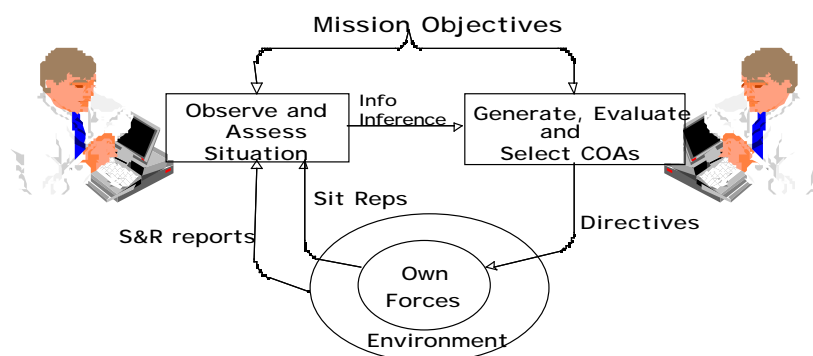


Figure 1. High Level Model of Major Command And Control Activities

This research concentrated on influence nets which are one type of probabilistic modeling. In particular, the software application, Situation Influence Assessment Module (SIAM) developed by SAIC [Rosen and Smith, 1996] was used. Design/CPN, was chosen as the target environment into which the influence net models would be converted by an automatic algorithm written using ML¹ code and many of the functions built into the Design/CPN application.

The remainder of this section provides a brief description of Influence Net modeling, motivation for the research, and a statement of the problem. Section 2 describes the conversion algorithm including design considerations and decisions. Section 3 provides an example that illustrates the process of course of action development and evaluation. The last section summarizes the results of this effort and includes a brief discussion of future directions.

1.1 What is an Influence Net

Influence nets are an extension to traditional Bayesian inference nets, sometimes called Bayesian belief nets or causal probability networks. Bayesian nets have both a graphical and a mathematical component and are used to model complex domains where uncertainty exists. They are directed acyclic graphs that represent the factorization of joint probability distributions of n random variables. They include an inferencing procedure for updating the joint probability distribution via Bayes rule as new information is received about any of the random variables. Thus they can be used as powerful modeling constructs for diagnostics and estimating outcomes given evidence. Because their construction requires the creation of marginal and conditional probability tables, they tend not to be easily accessible to decision makers not familiar with probability theory. Influence nets were developed to extend the traditional Bayes net structure and allow the creation of useful models by analyst unfamiliar with probability theory or who are unable to spend the time needed to fully specify a complete Bayesian net.

Influence net modeling incorporates an intuitive graphical influence diagramming technique for model construction. The SIAM implementation used in this investigation automatically creates a Bayesian model that allows for rigorous analysis using non mathematical inputs from the analyst. To construct an influence net, a modeler creates "influence nodes" that depict events that are part of a set of cause-effect relationships appropriate to the situation being modeled. The modeler also creates influence links between the nodes to graphically illustrate causal relationships between events. Each influence link can either be promoting or inhibiting, as identified by the terminator of the connecting link (arrow head for promoting, ball for inhibiting).

Figure 2 shows a simplified and hypothetical example of the topology of an influence net that might be created by intelligence analyst to assess a potential military situation and evaluate the effectiveness of a combination of diplomatic and military options. This net represents a situation where a Country B is in a posture that is threatening to its neighbors, creating instability. Country G is contemplating conducting a covert mission to destroy a key facility in Country B. If successful, the mission should reduce the confidence of the leader of Country B. The covert mission will be more effective if it can occur at the same time as a diplomatic mission visits a neighboring country. Additional influence will occur if the international community sanctions against Country B.

¹ ML stands for Meta Language. It is a functional programming language.

With SIAM, in order to generate the underlying quantitative marginal and conditional probability values, the modelers indicate, through a graphical user interface (GUI) the “strength” of each influence link to either promote or inhibit the effect event at the head of each link given the event at the tail. This approach is based on “Causal Strengths” logic [Chang, et. al. 1994] and the algorithm for converting causal strengths to marginal and conditional probability transition matrices to evaluate the cumulative likelihood of any event in the influence net using traditional probability calculations. Once constructed, influence nets can be used by analysts and decision makers to examine events over which they have some control to determine which events have the best chance of creating the outcomes desired by the decision maker.

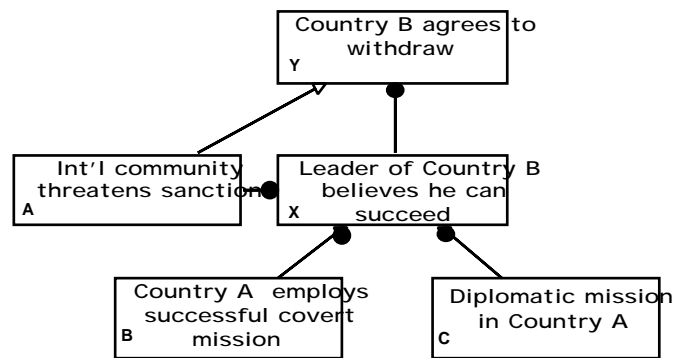


Figure 2. Topology of an Influence Net

1.2 Motivation for this effort

Influence nets produce static equilibrium models that relate cause to effect. There is no time or dynamic behavior captured in such models. They are most useful in assessing situations involving perceptions, rationale, and decisions. They are particularly useful if there is no underlying “physical” phenomenon that can be represented in analytical and executable dynamic models or if the underlying physical phenomena are not known.

One of the prime uses of influence nets is the evaluation of chains of causes and effects to determine which actions have the greatest likelihood of causing the desired outcome. In this sense, the influence net helps a decision maker decide *what* should be done. Detailing *how* the actions should be executed, in many cases, will involve the time-phasing of the actions. This can be modeled more effectively by a discrete event dynamical system. In many cases, such models require input parameters that can be estimated from influence nets. This effort was motivated by the notion of converting influence net models into an environment that can incorporate discrete event dynamical models so that both situation assessment models and dynamic models of courses of action can be interconnected to bring together the advantages of both modeling techniques into a single modeling environment. With such a process, it will be possible for analysts familiar with situation assessments to create influence nets in the environment with which they are most familiar and then to transfer their work in an automated way into an environment suitable for the execution of discrete dynamical models for evaluation of the selected courses of action.

1.3 Statement of Problem

The objective of this research was to design, develop, and test an algorithm that could take an output file from an influence net modeling software application and automatically generate an executable Colored Petri Net model. This model could then be interconnected to other executable CPN models. The Colored Petri Net model would perform the same forward probability propagation calculations that the influence net application performs. In addition, the algorithm should allow time delays to be added to the probability propagation at various nodes, allow easy input of marginal probabilities to the initial nodes for evaluating impact of different events, and allow the user to designate nodes for which a time history of change in probability could be collected for analysis.

2 Design of the Conversion Algorithm

In this effort, a UNIX-based influence net modeling application called Situation Assessment Module (SIAM) developed by Rosen and Smith (1996) was used. The design process started by understanding both the information available within a SIAM generated model and the calculations used by SIAM to propagate marginal probabilities given a change in probability of the initial events in the influence net. This understanding was used to design an output file that would be generated from SIAM and design a set of Colored Petri Net subnets that would replicate the probability propagation process. A multi step process for reading that file and converting it to a Colored Petri Net that would implement the forward probability propagation calculation was then developed.

This section first describes the SIAM processes and information and gives a brief description of the multi step process that accomplishes the conversion. This is followed by a more in depth description of the conversion process including descriptions of the Colored Petri Net that is created.

2.1 Influence Net Calculations and Export Information

The initial requirements for the design of the conversion algorithm focused on the probability calculations of SIAM that would be replicated in the Colored Petri Net. The calculation to be implemented is as follows.

Each node in the influence net represents some variable of interest and the variable is Boolean, in that it represents a logical statement that is either true (T) or false (F). Probabilities are associated with each node X expressed as a marginal probability $P[X]$. For each node in the influence net that has n parents ($n > 0$), let Q_i be the i^{th} set of parent states. A set of conditional probabilities, $P[X|Q_i]$, can be defined for each set of parent states of node X . SIAM uses the simplifying assumption that all the parent states are independent. With this assumption, letting $P[q_j \in Q_i]$ be the marginal probability of the j^{th} parent state in Q_i , then the marginal probability of that node X is

$$P[X] = \sum_{i=1}^{2^n} \left[P[X | Q_i] \times \prod_{j=1}^n P[q_j] \right]$$

$$q_j \in Q_i$$

- Case 1: $\{P(X|A, B, C, D)\}$
- Case 2: $\{P(X|\neg A, B, C, D)\}$
-
-
-
- Case 16: $\{P(X|\neg A, \neg B, \neg C, \neg D)\}$

Figure 3. Conditional Probabilities for Node with Four Parents

Given this calculation, the following information was needed from SIAM to create the conversion: for each node in the influence net, the name of the node and its identification number, its position in x and y coordinates on the diagram, the number of parents of the node, its initial marginal probability, and a vector of the $P\{X|Q_i\}$ representing the complete set of conditional cases for that particular node. For example a node X with four parents, A , B , C , and D , would have $2^4 = 16$ conditioning cases as shown in Figure 3.

For this effort, a special export routine was added to SIAM by Rosen and Smith that generates a text file with the above information for any influence net created in SIAM. Figure 4 shows an example of part of such an export files

2.2 Overview of the conversion algorithm

To meet the objectives of the procedure described in Section 1.3, a seven step algorithm was created in Design/CPN 3.0 which allows code to be written in the ML language to read files and create and draw CPN models based on the information in those files. The following briefly describes each step.

1. Read the text file from the Influence net application. For each node in the influence net initialize appropriate variables with:

" Caesar Baseline Net"	/ Net Title	node ID number
93	// Number of Nodes	node name
1	/ ID of Node @ X=1411 Y= 248	node location
" Country A employs SOF mission"	// Name of Node	conditional probabilities list
0.52025	// Marginal Belief of Node	list of parents
1	/ N number of Parents	node marginal probability
88	// Parent 1 "Country B selects options "	
0.00050	// (1 88)= (1 F)	
0.77650	// (1 88)= (1 T)	
2	/ ID of Node @ X=30 Y= 34	
" UN flies inspection flight"	// Name of Node	
0.66650	// Marginal Belief of Node	
0	/ N number of Parents	
3	/ ID of Node @ X=5 Y= 294	
" UN threatens sanctions"	// Name of Node	
0.66650	// Marginal Belief of Node	
0	/ N number of Parents	
4	/ ID of Node @ X=7 Y= 199	
" Leader of Country B believes he will succeed"	// Name of Node	
0.13845	// Marginal Belief of Node	
3	/ N number of Parents	
1	/ Parent 1 " Country A employs SOF mission"	
2	/ Parent 2 " UN flies inspection flight "	
3	/ Parent 3 " UN threatens sanctions	
president "		
0.93573	// (4 5,59,3)= (4 F,F,F)	
0.38379	// (4 5,59,3)= (4 F,F,T)	
0.80214	// (4 5,59,3)= (4 F,T,F)	
0.12467	// (4 5,59,3)= (4 F,T,T)	
0.01429	// (4 5,59,3)= (4 T,F,F)	
0.00141	// (4 5,59,3)= (4 T,F,T)	
0.00464	// (4 5,59,3)= (4 T,T,F)	
0.00046	// (4 5,59,3)= (4 T,T,T)	

Figure 4 Partial Sample of Export File

2. Create a page containing auxiliary boxes and arrows that replicate the topology of the influence net. For each node in the influence net, draw an auxiliary box and label it with the node name and ID number. Using the List of parents for each node, draw the arcs from parent nodes to child nodes. A sample of a replica of a 93 node influence net drawn with the algorithm is shown in Figure 5. For each node on the page, use the Design/CPN function that returns lists of input nodes and output nodes to generate a 3 tuple composed of 3 lists: a list of the node ID numbers of the parents (inputs), a list containing the node ID number, and a list of the node ID numbers of the children (outputs). These tuples of lists will be used in later routines to create the Petri Net.

3. Partition the net into sets of nodes that will be placed on individual pages in the Colored Petri Net.

4. For each page, create and interconnect subnets for each node of the original influence net that perform the probability propagation and updating for each node of the influence net.

5. Designate fusion places that interconnect each of the pages so that the topology of the Colored Petri Net is the same as that of the influence net.

6. Create a control panel page that allows the initial nodes of the net to be given marginal probability values and provides places to collect a history of the changes in marginal probability of nodes designated by the analyst.

7. Initialize the net by reading a file containing time delay information for each node.

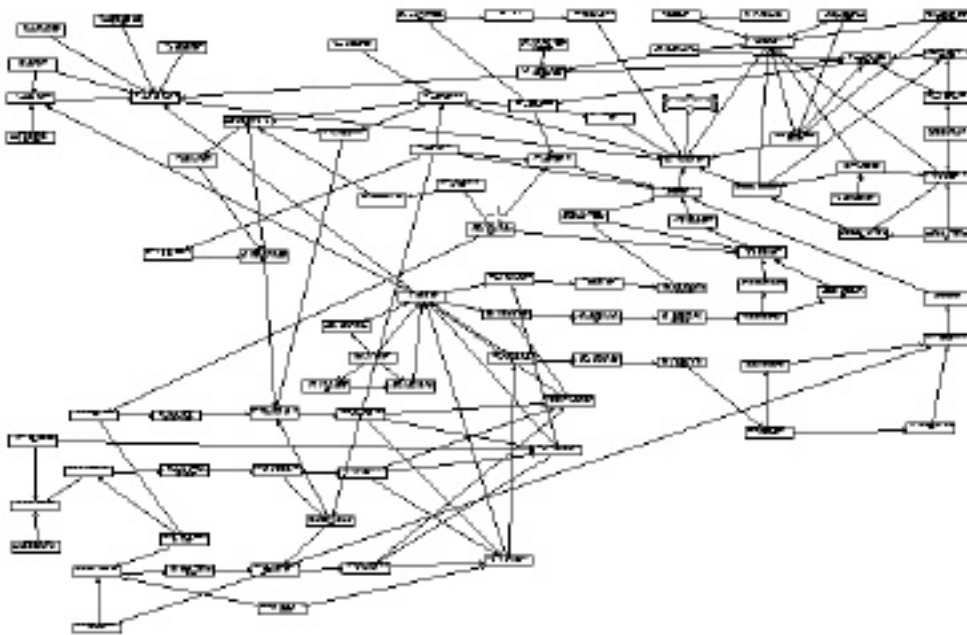


Figure 5 Sample Influence Net

The ML code is contained in several auxiliary boxes on a page in Design/CPN as summarized in Table 1. Each box of code is executed, one box at a time, until the complete CPN model has been created. At the end of each step, an appropriate dialog box appears indicating the step has been completed.

ML Code Boxes that Comprise the Conversion
1. Read the Input File; Construct Influence Net
2. Cluster Groups of Nodes for Paging
3. Draw Each Page of the Colored Petri Net
4. Create Fusion Place Ports and Sockets
5. Create a Control Panel Page
6. Initialize the Net with Timing Information

Table 1. ML Code Segments for Reading SIAM File and Building CPN Model

Once the Colored Petri Net has been constructed, a syntax check is made and the model is switched to the simulator mode. Once in the simulator, the model can be executed using different markings in the Control Panel Page that represent probabilities of the input nodes. The resultant history of changes in probability of the designated nodes is contained in the output places and can be saved as text files for analysis using a spread sheet program such as Excel. In addition to evaluating the influence diagram in the Colored Petri Net, existing Design CPN models representing the details of courses of action can be loaded into the appropriate pages of the influence net model and connected using fusion places.

2.3 Conversion Algorithm Design Details

Several constraints were encountered that had to be addressed in the design of the conversion algorithm. First, the current version of the influence net modeling application

(SIAM) creates one flat net on a single page. This is not a problem for small nets of less than 15 to 20 nodes; however, larger nets cause the equivalent CPN model to have an excessive number of objects on a single page in Design/CPN adversely affecting the run time of the simulation. The solution was to partition the CPN model into multiple pages that can be interconnected significantly decreasing the execution time of the CPN model.

Several schemes for accomplishing this partitioning were considered. To achieve the best performance, the goal of any partitioning scheme should be to create clusters of nodes that minimize links between clusters. Each cluster will be placed on its own page in the Colored Petri Net. Our initial design used a simplistic partitioning scheme by clustering nodes based on node identification number. For example, if a network had 60 nodes, and it was decided to partition the net into four pages, page 1 contained nodes whose identification numbers were between 1 and 15, page two contained nodes 16 through 30, etc. Since node numbers are assigned in a random fashion by the influence net modeling utility, it is not likely that this method will result in an optimal partitioning.

An algorithm to automatically create an effective clustering has not been written at this time as this was not the central focus of this research. In the currently implementation it has been left to the analyst to “read” the diagram and select the groups of nodes for each cluster. In general, humans heuristically can do a good job of creating a set of clusters that have a minimum or near minimal number of interconnections. At this time, the analyst determines the clusters and then enters the ID numbers of the nodes that are in each cluster. In future research we will implement one of several graph theoretic algorithms to do the clustering automatically.

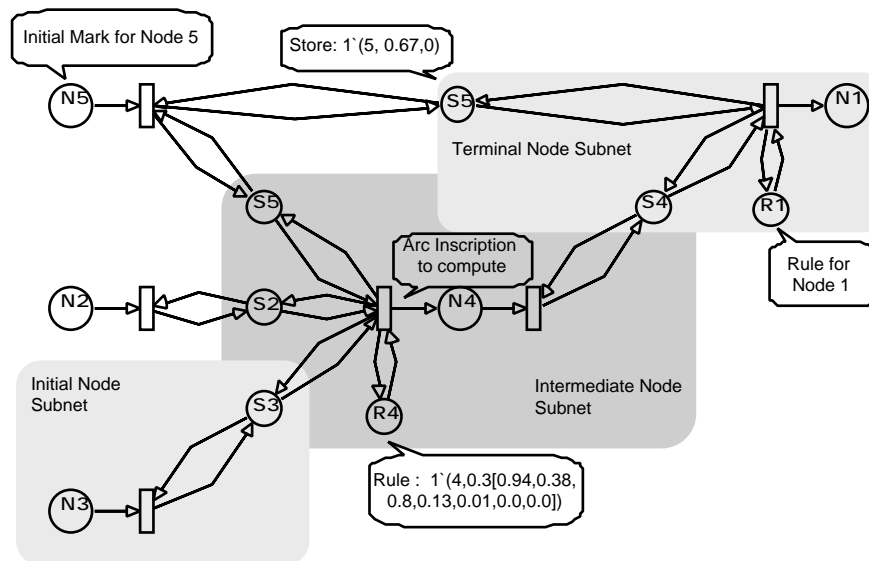


Figure 6. Structure of CPN of Influence Net

In step 4, the 3 tuples generated in Step 2 (they contain lists of parents and children for each node along with the lists indicating which nodes go on the different pages generated in Step 3) are used to generate the Colored Petri Net. Each node of the influence net is converted to one of three standardized subnets depending on whether the node is an initial node (one with no parents), a terminal node (one with no children) or an intermediate node (one with at least one parent and at least one child). Figure 6 shows the structure of the Colored Petri Net generated from a five node influence net with the topology of the influence net in Figure 2. It is composed of five standardized subnets, three for the initial

nodes (only the one for node 2 is highlighted) and one each for the intermediate and terminal nodes.

Figure 7 shows the CPN in more detail with the arc inscriptions and guard functions needed to understand how this structure executes. The Global Declaration Node for the CPN is shown in Figure 8.

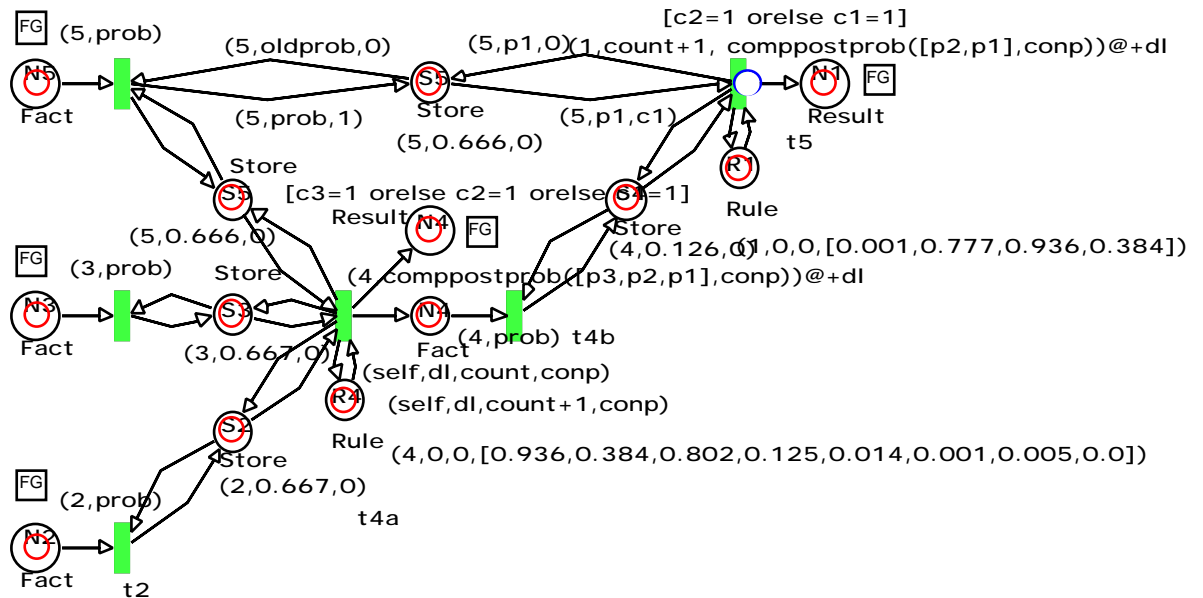


Figure 7 Annotated Color Petri Net

The Colored Petri Net uses four types of places. Places of the timed color set Fact contain tokens that are a tuple composed of node ID number (integer) and the probability (real). Places of color set Store hold tokens that are a 3 tuple of Node ID number, the probability of the node, and a control (integer). Places of color set Result hold tokens that are a 3 tuple of Node ID, a counter (integer) used to determine the sequence number of the token, and the probability of the node. Places of color set Rule hold a single token that is a 4 tuple composed of the node ID number, the time delay associated with that node, a counter, and a list of the conditional probabilities for that node.

In Figures 6 and 7, the places labeled N5, N2, and N3 (color set Fact) are initial nodes corresponding to nodes **A**, **B**, and **C** in the influence net of Figure 2. Each of their subnets is composed of a single transition with a single input place and as many output places as there are children of the node. These output places, labeled “S”, are of color set Store connected in a self loop for updating the marginal probability of the node. Figure 7 shows the arc inscriptions that remove the token with the “old probability” and replace it with a token with the new “probability.” Notice that when a new token is placed in the Store place, its control is set to 1.

Each intermediate node in the influence net is converted to a subnet structure with two transitions that are interconnected with a single place of labeled “N” of color set Fact. The first transition is connected in a self loop with the output Store places of each of its parents (so it can “read” the marginal probability of its parents). The arc inscriptions show that, when the transition fires, it removes the input tokens and replaces them with a token with the same probability but with the control set to zero. This first transition is also connected

with a self loop to a place labeled “R” (color set Rule) that contains a list of the conditional probabilities for that node and a sequence number. Each time the transition fires, it reads the conditional probability list, the sequence number and the time delay and returns a token with the same conditional probability list, and time delay and increments the sequence number by one. The output arc from the first transition to the N place has an arc inscription that computes the marginal probability of the node given the marginal probabilities of the parents and the list of conditional probabilities contained in the rules. The arc inscription uses the `computpostprob` function contained in the Global Declaration Node to perform the probability calculation contained in the equation in section 2.1. Its input arguments are the list of probabilities that are read from the parent Store places using variables `p1, p2,...` and the list of conditional probabilities contained in the rule using variable `comp`. The N place is timed and the arc inscription incorporates the time delay contained in the rule.

Because its input places always contain tokens, the first transition is enabled whenever the guard function evaluates as true, i.e., when at least one of the control variables of the input tokens is equal to one, meaning a new update has occurred. For example, in Figure 7, the guard function for the first transition of node 4 is `[c3=1 or else c2=1 or else c1=1]`.

The second transition in the intermediate subnet is connected in a self loop to a Store place for each child. Note that the second transition with its preset and poset have the same structure and arc inscriptions as the subnet for the initial nodes. Said another way, the structure of the initial node is the subnet of the intermediate node obtained by removing the first transition and its preset.

```
(* ===== Global Declaration Node =====*)

color FactID = int;
var f,self: FactID;
color Control = int;
var c1,c2,c3,c4,c5,c6,c7,c8,c9,c10: Control;
color Delay = int;
var td,dl: Delay;
color Counter = int;
var count: Counter;
color RealCS = real;
var prob,oldprob,p1,p2,p3,p4,p5,p6,p7,p8,p9,p10 : RealCS;
color RealList = list RealCS;
var comp,pli,problast: RealList;
color Rule = product FactID * Delay * Counter * RealList;
color Fact = product FactID * RealCS timed;
color FactTime = product FactID * RealCS * Delay ;
color Store = product FactID * RealCS * Control;
color Result = product FactID * Counter * RealCS timed;
fun constlist 0 = []
  | constlist n = (n-1)::constlist (n-1);
fun twopower 0 = 1
  | twopower p = 2*(twopower (p-1));
fun detbase 1 n = [n]
  | detbase p n =
    let val q = (n div (twopower (p-1)))
        in q::(detbase (p-1) (n-(q*(twopower (p-1))))))
    end;
fun probdet [] [] = 1.0
  | probdet (p::ps) (y::ys) = (if y=0 then p else (1.0-p))*(probdet ps ys);
fun multlist [] [] = 0.0
  | multlist ((x:real)::xs) ((y:real)::ys) = (x*y) + (multlist xs ys);
fun comppostprob(pli,problast)=multlist (map (probdet pli)
  (map (detbase (length(pli))) (constlist (length(problast)))))) problast;
```

Figure 8 Global Declaration Node

The subnet structure for each terminal node is similar to the structure of the intermediate nodes without the second transition (N1 is the only terminal node in Figures 5 and 6). There is a slight difference; the output place, which is of color set Result, contains a sequence number that ensures the list of output tokens are ordered in the sequence in which the transition fired.

Notice that the intermediate node subnet also has a Result place connected as an output of the first transition of that subnet. This place and its arc are automatically generated for each node designated by the user to collect time history data. It has the same arc inscription as the terminal node.

Before describing how the algorithm builds the Colored

Petri Net just described, it is instructive to understand how the net executes. As previously stated, the final Colored Petri Net contains initial markings for all Store places and all Rule places. The control element of each Store token is initialized to zero and the counters in the Rule tokens are initialized to zero. In the absence of any tokens in any Fact places, no transitions will be enabled. The CPN performs the forward probability propagation whenever a token is entered into a Fact place. This normally occurs in the initial node input places. When such a token is added, it sets off a chain reaction that characterizes the probability propagation. For example in the CPN of Figure 7, if a token $1(2, 1.0)@[0]$ is introduced into the N2 place, transition t2 will fire replacing the token in S2 with a token $(2, 1.0, 1)$. This will enable transition t4a because the control has changed to 1. Transition t4a will “read” all three probabilities of its parents and compute a new posterior probability. The result will be placed in the Result place with sequence number 1 and in the intermediate Fact place. If there is a time delay associated with this node, the token will be given the appropriate time stamp. When the simulation time advances, transition t4b will be enabled and fire, updating place S4. This will trigger the same action for node 1 as occurred with node 4. The result will be Stored in Result place 1. Of course several tokens can be placed in the initial input places to see the effect of several events. Each token precipitates a chain reaction like the one just described.

In Step 4, the algorithm that draws the CPN executes a sequence of actions. First, it draws the appropriate subnet for each node of the influence net. The subnet drawn depends on the 3 tuple that contains the lists of parents and children generated for each node in step 2. For example, for each intermediate node, a subnet like the one in Figure 9 will be drawn on the page in the same location as its counterpart in the influence net.

Next the algorithm draws the Store places, one for each arc in the influence net drawn in step two. If two nodes are on the same page, the Store place is drawn halfway between the first transition of the parent and the second transition of the child. Then, one pair of arcs with arc inscriptions are drawn between the Store place and the second transition of the parent place and a second set of arcs is drawn between the Store place and the first transition of the child. If the two nodes are on different pages, the Store place of a parent is drawn at a fixed location near the first transition and is designated as a fusion place. Arcs are then drawn between the first transition of the child node and this Store place. Note that the second set of arcs is not drawn at this time. This procedure is completed for each page until all nodes have been constructed.

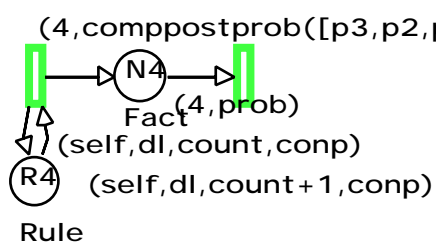


Figure 9. Initial Subnet for Intermediate Node

In Step 5, the algorithm uses the list that associates each node with its children to connect the pages together using fusion places. This only needs to be done when a parent and child are on different pages. Starting with the first page, the algorithm searches until it finds a node with a child on a different page. It then creates a Store place near the second transition of the node subnet and adds it to the group of

fusion places that was created on the child’s page when that fusion place was created in Step 4. The algorithm continues until all nodes have been searched and, as a result, all fusion places appropriately grouped.

In step 6, the algorithm creates a separate control panel page. The control panel page enhances the usability of the network by providing a convenient single input place to input

the probabilities of initial nodes in the net and places to collect the output data of any node in the network as designated by the analyst². The control panel page for the CPN in Figure 6 is shown in Figure 10. All of the places are fusion places. The box in the middle is an auxiliary node that serves no function other than indicate the relationship of the control panel page to the other pages in the Colored Petri Net. The figure shows the form of the output data that is collected in each Result node. Notice that the listing of the tokens is ordered by sequence number so that it gives a time history of the probability state of the node on which the data is collected.

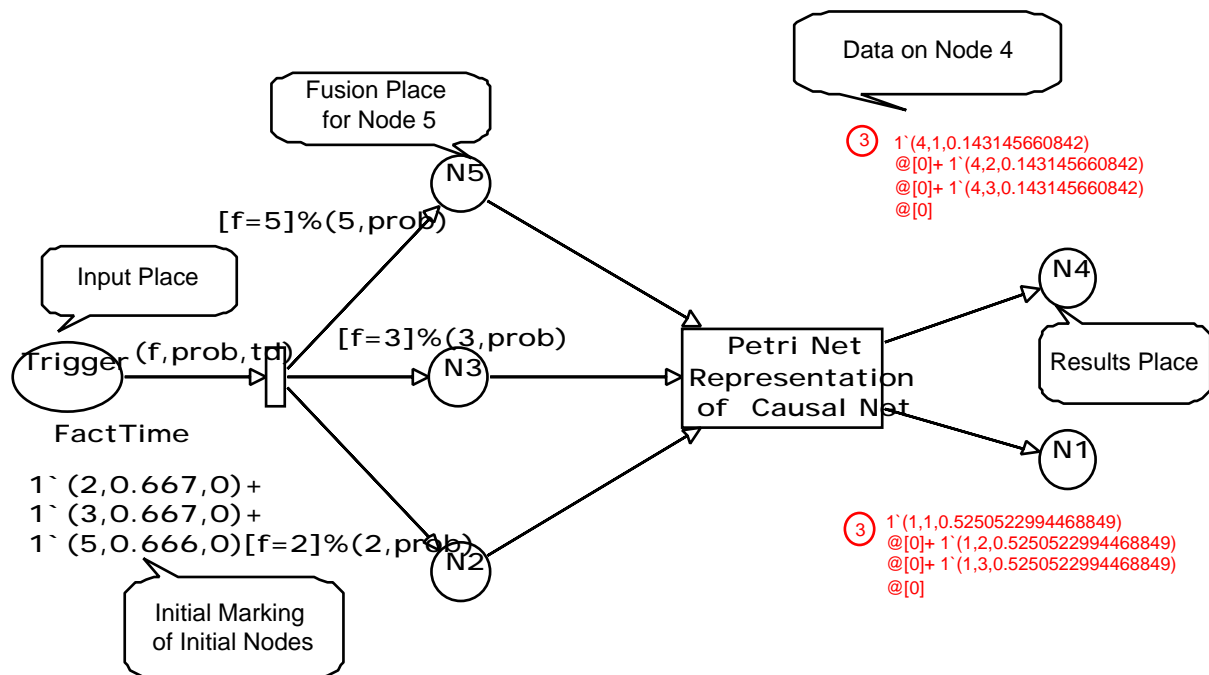


Figure 10. Control Panel Page

To create the control panel page, the algorithm searches each page for initial nodes and designates their input places as fusion places. It also searches for Results places and designates them as fusion nodes. The algorithm then creates a new page and creates Fact places for each input node and adds them to the appropriate group of fusion places. It performs a similar operation for the Results places. The algorithm creates the Trigger place (color set Fact Time) and the transition shown in Figure 10, and draws the appropriate arcs. Finally, an auxiliary box is created and auxiliary connectors are drawn between each fusion place and the auxiliary box.

In the last step, the algorithm creates the initial marking regions for the entire network. It first provides a dialog box asking the user to navigate to and select the text file that has the time delays associated with each node in the influence net. After reading this file, it creates the initial marking regions for all the rule places using the variable values assigned when the SIAM export file was read in Step 1. It also creates the initial marking regions for every Store place in the net. Finally, it creates an initial marking region for the Trigger place on the control panel page by creating a list of the marginal probabilities for the initial nodes and converting it to an initial marking like the one shown in Figure 9.

² The analyst can “instrument” the net by designating the nodes from which a history of the probability changes will be recorded. An additional instrumentation place is automatically added to each subnet of these nodes to store status tokens each time a node is updated.

2.4 Difficulties encountered and overcome

The development of the algorithm was initially accomplished on both Motorola 68030 CPU and PowerPC Macintosh computers using Design/CPN 3.0. The code was then ported to a SUN Sparcstation Ultra 1 with little difficulty.

During the development, several difficulties with Design/CPN were encountered for which work arounds were devised.

As was mentioned in Section 2.3 we found that when using any influence net of reasonable size, the net had to be partitioned to decrease the simulation run times. We also found a significant difference in performance with different computers. For example, the time to perform the conversion with a Power Macintosh 7500/100 was approximately 2 hours compared to 25 minutes for the SUN Ultra 1 for a 93 node influence network. We found a similar 4 to 1 improvement on execution of the simulator for the same net.

One difficulty occurred when running the application on the PowerPC Macintosh. It is our conjecture that there is a limitation on the maximum number of tokens that can be present in any one place. This limitation was exceeded in Macintosh versions of the algorithm when using an influence net containing 93 nodes and 23 initial places. The Macintosh version of Design/CPN allows one to observe the running of the ML code. We were unable to do this in the UNIX version. Observation of ML code revealed error messages of "out of range" once the number of tokens exceeded approximately 900 in the single terminal place of the 93 node net; thus our conjecture on the maximum number of tokens.

The design relies on a great deal of information encoded on the tokens. In particular, the size of the Rule token grows exponentially with the number of parents. We had several nodes with 8 or 9 parents. This generates lists of conditional probabilities that have 256 or 512 real numbers. We conjecture that some of the problems we had occurred because we exceeded memory limitations in these tokens. The problems disappeared when we used a rounding function to hold the precision of probability calculations to three significant digits. The large number of characters in the string of rules also makes the viewing of large networks difficult. Figure 11 shows a portion of a page for a 93 node net with many of the arc inscriptions suppressed.

One problem that was difficult to overcome occurs when saving the state of the CPN model in the simulator. If each page of the CPN model is designated as a prime page and then the state is saved, the resulting file will contain a corrupted multiplicity factor of minus 5147 for several of the pages. Of course the model will not execute properly. The reason for this anomaly is not known. The following work around, suggested by Kjeld H. Mortensen of the University of Aarhus, Denmark, eliminated the problem. When the model is created in the editor, only the first page of the net is designated as prime. After switching to the simulator, the state can be saved and reloaded without any problems. Of course after loading the state, all of the pages have to be designated as prime. The initial state command is then issued using the CPN pull-down menu, and after the re-initialization that now includes all the pages, the model can be executed to collect data for analysis.

3.0 Example

The following example illustrates the procedure for creating an influence net of a politico-military situation, using it to select high pay-off actionable events, converting the net to a CPN, and executing the CPN to collect data for course of action evaluation.

Witmania are also included. The intelligence analysts also include several military actions, including the use of a covert mission to destroy the WMD production facility, the use of a Unmanned Air Vehicle (UAV) as a diversion, and a neutral flight. The influence net models the reaction of the Witmania integrated air defense system (IADS) to the covert operation, UAV and neutral flights. The overall result is a large influence net created in SIAM with over 90 nodes, including 23 nodes for actionable events. Figure 3 in section 2 is a thumbnail view of the structure of the influence net.

The intelligence analyst conducts a sensitivity analysis of the influence net to determine which controllable events have the greatest impact on the target node, stopping the WMD program. This analysis indicates that while diplomacy has some impact, the combined effects of the covert operation, UAV, and neutral flight significantly increase the likelihood that the WMD program will be stopped. As a result, it is decided to investigate combinations of these actions in more detail. Figure 12 shows a notional operational concept for the various courses of action under consideration.

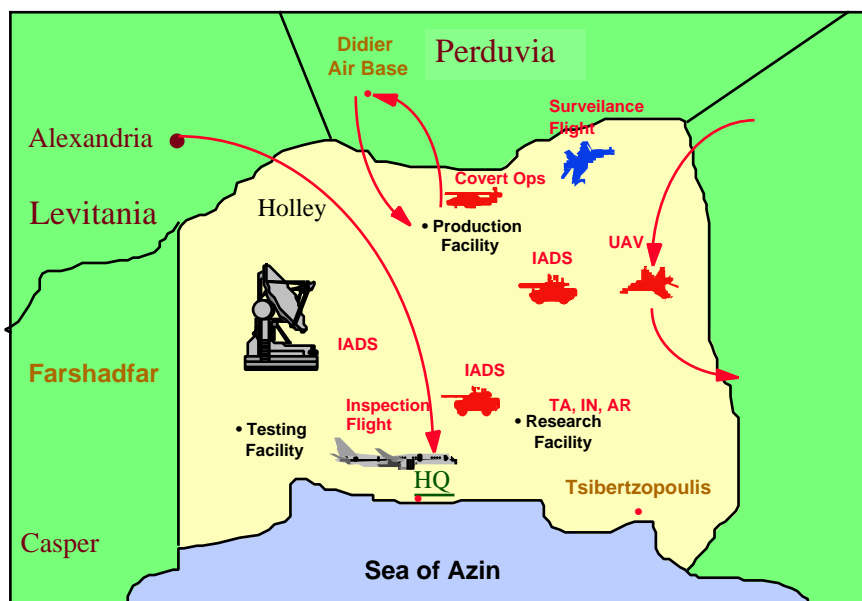


Figure 12. Operational Concept for Witmania

Using the output file from SIAM and the conversion algorithm created in Design/CPN the influence net is converted to a colored Petri Net. Figure 11 shows a portion of one of the pages of the CPN. First, analysis of the sequences is conducted to eliminate any sequences that will not have the desired impact. Then the timing information is added to the best sequences to analyze the temporal aspects of the courses of action.

It was decided to include the diplomatic events in all courses of action because these actions would take place prior to military action and may produce the desired results. The diplomatic efforts are composed of a sequence of four actions: Initial diplomatic efforts, surveillance flights, using the results of the surveillance flight to advertise the existence of the WMD program to the international community, and local psychological operations to influence public opinion against the WMD program. Figure 13 shows how the probability that the WMD program will be stopped increases from 26% to 48% as this sequence of diplomatic actions takes place.

Since the sensitivity analysis using SIAM indicated that the target node probability would increase significantly if a covert mission and UAV flight were conducted in conjunction with a neutral flight, the CPN model is executed using the complete set of 13 combinations of sequence of these actions shown in Table 2.

Neutral Flight	UAV	Covert Mission
1	1	1
1	1	2
1	2	2
1	2	1
1	2	3
1	3	2
2	1	1
2	1	2
2	2	1
2	1	3
2	3	1
3	1	2
3	2	1

Table 2. Sequences for 3 Actions

The control panel page of the CPN model has been set up to collect probability data on several intermediate nodes in addition to the target node. These include the probabilities that the Integrated Air Defense System (IADS) stands down due to the neutral flights, that the UAV survives, that the covert mission evades the IADS, and that the overall cover mission is a success. Figure 14 shows the results for the third sequence of Table 2: the neutral flight followed by the covert mission concurrently with the UAV. Similar charts are created for the 12 other sequences. The 13 charts show that the covert mission success dominates the outcome of the target

node. The neutral flight increases the likelihood that IADS will stand down and the UAV has a 60% chance of survival. The overall conclusion is that the UAV provides little cover for the covert mission; covert mission success is independent of the UAV flight.

After reducing the number of candidate courses of action by analyzing the impact of the various sequences of actions, timing data including processes times for the various nodes in the net and starting times for the actionable events in the CPN model, is added to the model. For example, estimates of the time delays in passing information through the

Decision to Stop WMD Program

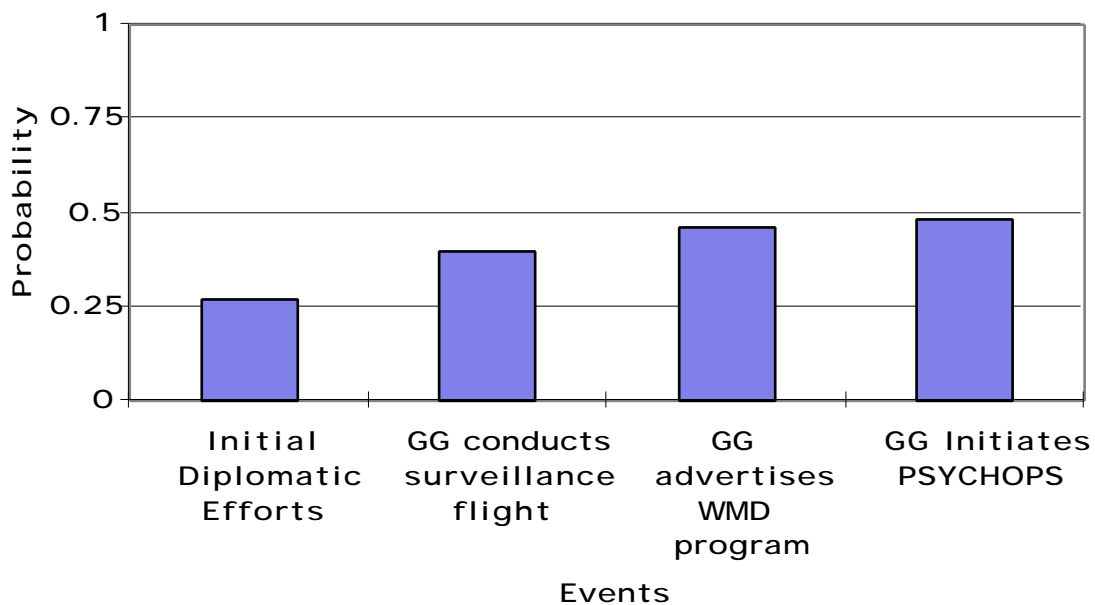


Figure 13. Changes in Probability of Target Node Due to Diplomatic Actions

IADS and the time required for decisions to be made to stand down the IADS or have the IADS engage the covert and UAV missions are added to the appropriate nodes of the CPN.

In addition, the planned time for the flight of the covert mission, the UAV and the neutral flight are incorporated in the tokens representing these events. The model is executed again for several combinations of these event times and data collected on the key nodes of the model.

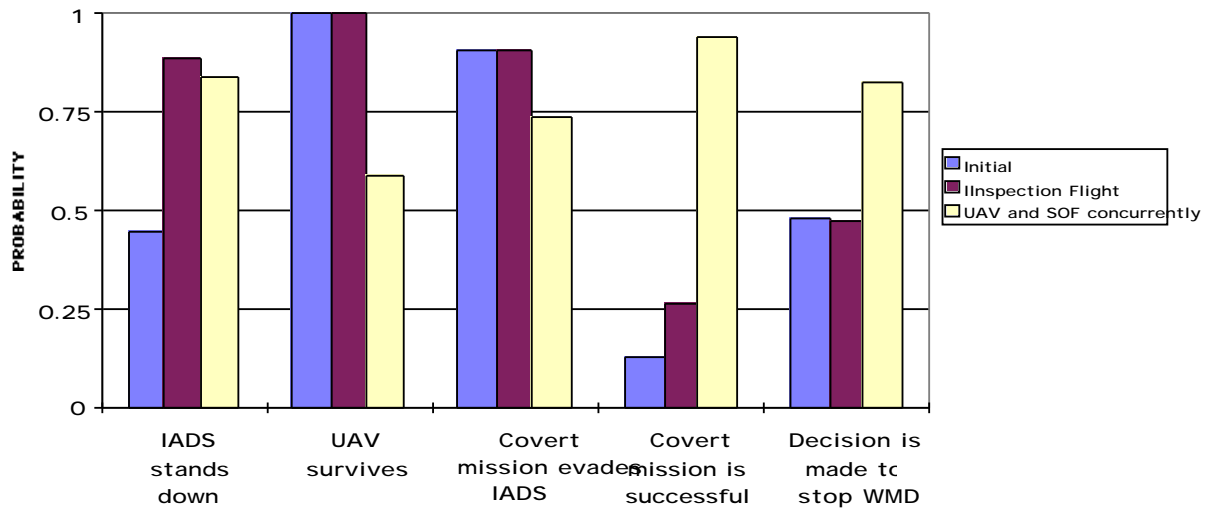


Figure 14. Results of Inspection Flight Followed by SOF and UAV

The data collected indicate the time phased changes in probability of each event for which data was collected. Figure 15 shows an example to the changes in probabilities that the IADS will stand down due to the neutral flight and that the UAV will survive. Note that the advent of the neutral flight causes the probability of the IADS standing down to increase. As the UAV enters the IADS air space its probability of survival begins to decrease until approximately 25 minutes into the mission. Then there is a slight increase due to the interaction with the neutral flight. As the UAV mission lingers in the air space beyond 35 minutes, its probability of survival again decreases. This suggests that if the UAV is to be used, it should be kept in the IADS airspace for no more than 35 minutes. Other time histories of the probabilities of events are examined until the best time windows are determined.

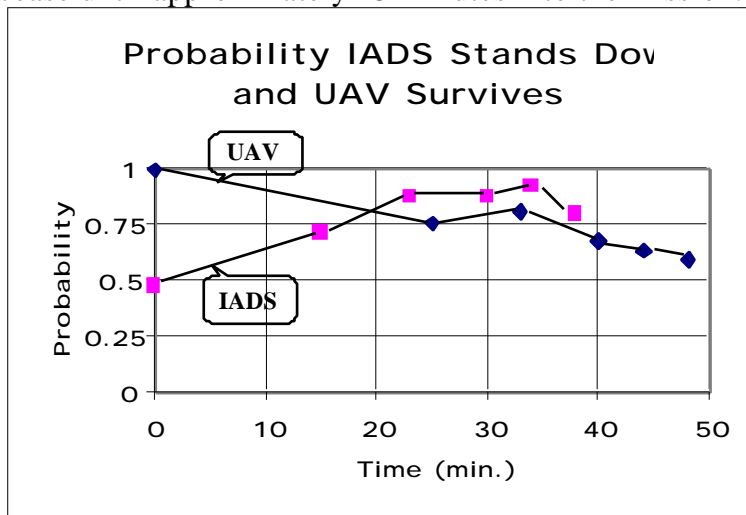


Figure 15. Timing Comparison of Two Events

4 Summary and Areas for Further Research

This work represents a successful integration of influence nets with discrete event models in a common environment. It is believed that this technology will allow a closer coupling between models designed to assess situations and compare potential courses of

action, and dynamical models that can be used to provide detailed planning and evaluation of those courses of action.

To date we have not connected together any real operational models and influence nets in this common environment. Future research is planned to do this and develop guidelines and rules for constructing and interconnecting such models that is consistent with both the underlying mathematics of the models and the objectives of the enterprise. In addition, a sound methodology using the interconnected models for developing and evaluating courses of action based on measures of performance and measures of effectiveness for the mission objectives needs to be developed. When fully established, this capability will enhance understanding of collaboration between teams of experts, each using tools and models appropriate to its discipline for situation assessment and course of action generation and evaluation.

Acknowledgements

This work was sponsored, in part, by the United States Air Force Research Laboratory (George Mason University participated as a sub contractor to QuesTech, Inc.) and by the Office of Naval Research under contract no. N0014- 93-1-0912. Special thanks goes to Mr. Steve Hendricks and Mr. Larry Simmons of QuesTech for their contribution in defining the hypothetical scenario used in this effort, and to Dr. Julie Rosen and Mr. Wayne Smith of SAIC for there assistance with SIAM.

References

[Buede and Wagenhals, 1996]. Dennis M. Buede and Lee W. Wagenhals. *Influence Diagram Representation of Dynamic, Distributed Decision Making*, Proc.of the Command and Control Research and Technology Symposium, Naval Post Graduate School, Monterey CA, June 25-28, 1996.

[Chang *et al*, 1994] K.C. Chang, Paul E. Lehner, Alexander H. Levis, S. Abbas K. Zaidi, and Xinhai Zhao. *On Causal Influence Logic*, Technical, George Mason University, Center of Excellence for C3I, December 3, 1994.

[Rosen and Smith, 1996] Julie A. Rosen and Wayne L. Smith. *Influence Net Modeling with Causal Strengths: An Evolutionary Approach*, Proceedings of the Command and Control Research and Technology Symposium, Naval Post Graduate School, Monterey CA, June 25-28, 1996.