Abstract

We present an environment for web-based simulation of influence nets to be used for operational planning. Previous work in this field has shown how influence nets, which are used for probabilistic modelling, can be extended with time and then translated into a coloured Petri net to do temporal evaluation of plans. Simulating the coloured Petri net model in a web environment makes it easy for subject matter experts to use the simulator for planning without knowing the underlying influenced Petri net formalism and tools. This paper discusses the use of influence nets for operational planning, a simulator for influence nets implemented using coloured Petri nets, and the architecture of the complete web-site which can be used for operational planning.

Keywords: Operational planning, Courses of action, Effects based operations, Influence nets, Coloured Petri nets, Web-based simulation.

1 Introduction

Planning an operation to achieve objectives can be a very complex task. A plan may depend on interplay between several different complex events, where only some of the events are controllable. In many cases, plans are developed to try to compel an adversary to take actions or make decisions that the adversary is not pre-disposed to make.

The operational planning method presented in this paper has been motivated by a concept called effects based operations (EBO). EBO is the notion of selecting actions based on their collective contribution to desired and undesired effects. The set of selected actions is called a course of action (COA). To support EBO, at least two problems must be addressed. The first is to relate effects to actionable events. In this problem, we need to define the set of desired and undesired effects. The second problem is to consider constraints associated with specific actions or combinations of actions, the selected actions must be sequenced and time phased. This timing of the actions is important to be able to know which order and when the controllable actions should be done to obtain the best possible conditions for a successful operation. The result is a set of alternative COAs. These COAs are then evaluated against requirements to determine the COA that provides the best likelihood of causing the desired effects to occur and the undesired effects to not happen.

Probabilistic models like, e.g. Bayesian inference nets (Jensen 1996) and influence nets (Rosen & Smith 1996), can be used to model a situation to determine which actions to take to optimise the outcome of an operation. However, these models do not include the temporal aspect of the planning.

Previous research on influence nets is presented in, e.g. (Wagenhals, Shin & Levis 1998) and has demonstrated how timing information can be added to influence nets by translating the influence net and a timing profile to a coloured Petri net (CP-net or CPN). This CP-net can then be simulated both to estimate the probability of a successful operation, and to evaluate a plan for timing the operation. The translation from the influence net and the timing profile to the CPN model is done completely automatically, i.e. the places, transitions, and declarations of the CPN model are generated without user interaction. In this paper we refer to a CPN generated using this method as a fixed CPN model.

The influence net formalism has been extended slightly to include so-called logical gates. An influence net with logical gates together with timing information is in the following called a timed influence net with logic (TINL).

Recent work, presented in (Lindström & Haider 2001), has developed a generic CPN model that is able to simulate any TINL. In other words, a TINL CPN simulator has been implemented. The advantage of the generic CPN model is that time is saved during the translation, because a new CPN model does not need to be generated each time a new TINL needs to be examined as it had to when generating fixed CPN models. Instead, the generic CPN model within the TINL CPN simulator is initialised with appropriate tokens to reflect the concrete TINL. One issue of concern was that the simulation time for the generic TINL CPN simulator would be much longer due to the complex colour sets and potential large number of tokens in places required by the generic CPN model. Experiments presented in this paper has shown that using a more effective CPN simulator has overcome this problem.

It is essential that tool support is available to and usable by (1) the team of intelligence analysts and other subject matter experts (SMEs) responsible for analysing a situation in terms of actions and effects...
and (2) operational planners who perform resource allocation and scheduling to cause actions to occur. A modelling tool that can be used by these teams called CAESAR II/EB (Wagenhals & Levis 2001) has been built to focus on the belief and reason aspects of an adversary so that potential actions can be related to effects. The tool incorporates influence nets as the probabilistic modelling technique and the TINL CPN simulator to support the temporal aspects of COA evaluation. These two formalisms enable the modeller to create the structure of actions, effects, beliefs, decisions, and the influencing relationships between them.

A two stage operational concept that uses the two modelling techniques to perform COA analysis has evolved through tests in realistic scenarios (Wagenhals & Levis 2001). Figure 1 illustrates the two stages in applying the CAESAR II/EB tool. In the first stage, intelligence analysts and SMEs develop an influence net to specify the probability of effects given sets of actions. Once the influence net has been created, it is exported to an influence net specification file so that operational planners can perform temporal evaluation in stage two. The goal of stage two is to determine and recommend the timing of the set of actions that give the best set of acceptable probabilities for all effects. In stage two, a delay file for the influence net is created by the operational planner to specify a full TINL. Then temporal analysis can be conducted to analyse different COAs by simulating the TINL using the TINL CPN simulator. The simulation results provide, for a given timed sequence of actions, the probability of effects over time, in the form of a probability profile (Wagenhals & Levis 2000). Probability profiles indicate how long it will take for a specific COA to achieve the desired effects, reveal time windows of risk when the probability of effects is unacceptable, and provide time windows for indicators of success or failure. Changing the timing of selected actions can significantly change the probability profiles.

Thus, a tool like CAESARII/EB can be used to support an overall planning process in which a situation analysis team creates a model with the tool that relates potential actions to overall effects that can be used by a team of operational planners that perform the resource allocation and scheduling function to evaluate plans. The operational planners execute the model using the tool to evaluate proposed plans for acceptability and may adjust the timing of actions within resource constraints until acceptable outcomes in terms of the timed phased probability of effects is achieved.

One problem of translating an influence net to a CP-net is that the analysis is conducted using another formalism than the one which is used for specifying the influence net. In practice, neither the intelligence analyst nor the SME necessarily knows the CPN formalism or the CPN tools. Therefore, the translation puts an extra, unnecessary skill requirement onto the SME. As a consequence, an alternative domain-specific graphical user interface (GUI) for the TINL CPN simulator has been developed, and is presented in this paper. The GUI is created using simple so-called HTML documents and CGI scripts (Gundavaram 1996), and is based on the tool Design/CPN (DesignCPN 2002) and on the method presented in (Lindstrom 2001). Simulating the CPN model using a domain-specific GUI makes it easy for the SME to use the simulator for planning without knowing the underlying CPN formalism and tools. The main objective of this paper is to illustrate how to create and use a domain-specific GUI in a web-based environment for simulating TINLs to be used for operational planning.

The paper is structured as follows. Section 2 describes how influence nets and the TINL CPN simulator can be applied in practice for operational planning. Section 3 discusses the generic CPN model. Section 4 describes the web-site used to control the TINL CPN simulator. Section 5 compares the speed of executing alternative CPN models that have been proposed for simulating TINLs. Finally, Sect. 6 concludes and gives directions for future work.

2 Practical Use of the Operational Concept

In this section we describe the two stages of the operational concept. First we describe stage 1 by introducing influence nets and discussing how an influence net can be created to support effects based operations. Next, in stage 2 the execution of an influence net using the TINL CPN simulator from the web environment is described.

2.1 Stage 1: Creating Influence Nets

Influence nets, a variant of Bayesian nets, have been used since 1994 (Levis 2000, Rosen & Smith 1996) to depict the causal relationships between actions and events. They are acyclic directed graphs. A small example of an influence net with six nodes is shown in Fig. 2, and will be used for illustration in this paper. The modelled situation is that Fed wants to set conditions so that Borg will not use the weapon of mass destruction called a death star (DS) against Fed in a war. The leftmost nodes in an influence net represent actionable events which are either desired or undesired – seen from the point of view of the SME. In the middle are the nodes relating actions to effects.

The nodes in an influence net represent statements or beliefs with which a probability value can be asso-
associated. For example, the upper-middle node 26 represents the statement that “Borg believes use of strat DS is counterproductive”. Each node with parents has a parameter called the baseline probability which indicates the probability that the event can be true independently of all other modelled events. The baseline probability is positioned in the lower right corner of the nodes with parents in an influence net (0.5 for node 26).

A directed arc represents a directed binary relationship between two nodes. Two numbers, called influences, characterise the relationship and are noted by $h$ and $g$. The first one, $h$, represents the strength of the influence that a parent node has on a child node, if the parent node was to be true. The second one, $g$, reflects the strength of the influence if the parent node was not true. Both $h$ and $g$ can take values in the closed interval $[-1, 1]$ which means that the binary relations can be either promoting (+) or inhibiting (−). For example, the SME believes that node 25 (upper left) has impact on node 26. In the actual application, the analyst generally uses a set of qualitative statements that map to a set of values in the interval $[-1, 1]$. The typical statements and their corresponding values are as follows. Significantly less likely: -0.99, moderately less likely: -0.67, slightly less likely: -0.33, no effect: 0, slightly more likely: 0.33, moderately more likely: 0.67, significantly more likely: 0.99. For example, to set an $h$ value, the analyst will select an answer from the question: if the parent were to be true, the impact on the child would be to make it e.g. “slightly more likely” (select the best statement from the list). In the example, the analysts estimate is that when node 25 is true, then the node 26 will be slightly more likely to be true (thus $h = 0.33$), and when node 25 is false, then node 26 will be slightly less likely to be true ($g = -0.33$).

After an analyst has created an influence net and assigned the values of the $g$, $h$, and baseline probability parameters throughout the net, the values are translated into conditional probabilities for each node with parents using an algorithm called CAST (for Causal Strength) (Chang, Lehner, Levis, Zaidi & Zhao 1994). The influence net can then be used to propagate probabilities from the leftmost nodes with no parents (action nodes) to the rightmost nodes with no children (effect nodes). In this paper we have used the CAT-tool\(^1\) to create influence nets. In addition, the CAT-tool has been used to automatically convert the influences ($h$’s and $g$’s) between parent and child nodes to the conditional probabilities via the CAST algorithm.

Once the influence net has been completed, it can be used to evaluate the impact of actions on the effects (decisions) of interest. This can be accomplished by sensitivity analysis or by executing the influence net. The sensitivity analysis consists of finding those actionable events that have most impact on the effects of interest. This analysis makes it possible to use only those actionable events which have the most impact on the effect nodes. To execute the influence net, the analyst sets the probabilities of a set of actionable (leftmost) events to either zero or one, depending on whether the action is planned or not, and evaluates the influence net. Then the tool propagates these probabilities from left to right until all effects are accounted for in the rightmost nodes representing main effects. An analyst can experiment with the influence net by changing the probabilities of one or more of the actionable events and seeing what the effect is on the key decision nodes.

Once the specification of the influence net has been completed, and the actionable events have been selected then the influence net is exported from the CAT tool to a so-called influence net specification file. This file is a textual representation of the influence net with the static information which is necessary to simulate an influence net. The file is then transferred to a web server where it is used by a script to specify the initial markings of the generic CP-net within the TINL CPN simulator.

### 2.2 Stage 2: Temporal Evaluation using CP-Nets

The next activity involves the operational planners who assess the availability of resources to carry out the tasks that will result in the occurrence of the actionable events. The resulting plan will indicate when each actionable event will occur. Selecting the set of actions that will lead to achieving the overall desired effects while not causing the undesired ones is not the only important task. Determining the timing of these actions is critical to achieving the desired outcomes. It is not possible to evaluate the impact of the timing in the CAT-tool because influence net does not contain temporal information. However, because influence nets assume the independence of causal influences, it is possible to associate time with either the nodes or the arcs of the influence net. The time delays represent the amount of time it takes for knowledge about a change in the status of any node to be propagated by some real world phenomenon to the node that is affected by that change. The update in the marginal probability (the current probability) of a node occurs immediately after the time delay. It is these time delays, along with the timing of the actions, that causes the generation of probability profiles which are graphs depicting the probability of each node as a function of time.

In this paper we will consider only two types of temporal information. The first one is the timing of the actionable events, i.e. the specification of when the controllable actions should be carried out. The second type of delay is the estimated time it takes before the consequence of each event represented in the influence net can be seen. This second timing information is expected to be constant for several COAs. An example of these two types of timing information is that the event represented by node 25 in Fig. 2 may be initiated after two hours and then the effect of the event will be seen after three hours.

The input scenario is described in terms of the actions chosen from the set of actionable events, in the selection of the COA and the time at which these actions occur. The actions are modelled as events, which means that they occur instantaneously. To give an idea of different scenarios, an example with two COA scenarios is shown in Fig. 3. The actions and their timing of COA1 are indicated below the time line while the same set of actions with different timing that comprise COA2 is shown above the time line.

Let us now turn to how the CAESAR II/EB tool is used for temporal evaluation. An analyst uses a

---

\(^1\)CAT is the Effects Based Campaign Planning and Assessment Tool developed by the US Air Force Research Laboratories (AFRL/IF) and George Mason University.
Figure 4: Web page to specify time delays.

```
<table>
<thead>
<tr>
<th>Node name</th>
<th>Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fed forces are in position to strike Borg DS</td>
<td>1</td>
</tr>
<tr>
<td>capability_28</td>
<td></td>
</tr>
<tr>
<td>Fed NCA pledges not to use DS first_25</td>
<td>2</td>
</tr>
<tr>
<td>Fed press reports major demonstrations in Fed</td>
<td>1</td>
</tr>
<tr>
<td>against war_29</td>
<td></td>
</tr>
<tr>
<td>Borg believes use of strat DS is counterproductive_26</td>
<td>3</td>
</tr>
<tr>
<td>Borg believes use of DS will push Fed to</td>
<td>1</td>
</tr>
<tr>
<td>negotiate_30</td>
<td></td>
</tr>
<tr>
<td>Borg decides to use DS option_27</td>
<td></td>
</tr>
</tbody>
</table>
```

Figure 5: Web page to specify a COA and to select nodes for display of probability profile.

The web browser to specify the temporal information and to simulate the influence net under a variety of initial conditions. This is done by filling out a set of HTML forms that are automatically generated using a so-called CGI script (Gundavaram 1996). A CGI script is a program that runs on a web server, and it can be started, for example, by submitting an HTML form from a web browser.

First, the temporal information must be specified by the analyst. The influence net specification file from the CAT tool is used by a CGI script to automatically generate an HTML document, where the operational planner can input the time delays of the individual nodes. Figure 4 contains such an HTML document for the influence net in Fig. 2. The HTML forms contains an entry for each node in the influence net with the text from the corresponding node in the influence net. When the form is submitted, the information is saved, and can later be accessed using the delay name specified in the form.

Once the time delays have been specified, the operational planner is ready to analyse different COAs using the TINL CPN simulator. This is done by first requesting the generation of a new HTML document where the input scenario can be specified. Figure 5 contains an example of such a document. The document contains two forms where input can be given. The first form is for specifying the timing for the actionable events that should be included in the COA. The second form is used for two things. First, the operational planner must specify if there is strong evidence that some of the events are already true. For those events, the effect is set to 100% in the form. Second, it is specified for which of the nodes, the probability profile should be included in the graph of the probability profile.

When the HTML document is submitted, the web server sets the initial marking of the generic CPN model within the TINL CPN simulator and simulates the CPN model. During the simulation, the server automatically stores the values needed to plot the probability profiles in a file for later use in comparing COAs. When the simulation ends, graphs are generated which contain the probability profiles that show the marginal probability for the selected nodes in the influence net as a function of time, and are then displayed in the web browser. The probability profiles indicate how long it will take for the effects of the actionable events to affect various nodes in the influence net and time windows when probabilities may have unacceptable values. By changing the timing of the actions in the COA, the analyst may be able to eliminate these unacceptable windows. The analyst will most likely concentrate on the probability profiles of the key decision nodes, i.e. the nodes with no children.

An example of three probability profiles for a single COA is shown in Fig. 6. The graphs contains plots of an extended version of the influence net for the Borg-example. The annotation boxes in the plot have been added manually for clarity. For this COA, the probability profile for node 30 indicates that after 5 hours Borg wants to negotiate with a probability of about 62%. The likelihood that Borg will decide to use weapons of mass destruction (WMD) decreases and then increases before it finally reaches a very low value. This indicates that there is some risk associated with this COA. The analyst will attempt to find an alternative timing scheme that will reduce this risk caused by the rise in the likelihood of WMD use.

To compare multiple COAs the analyst can fill out another HTML form in the web browser to generate plots that show the probability profiles of nodes for different COAs. Figure 7 contains an example of such an HTML document for the Borg example. In the first form a single node is selected, and two COAs are selected in the second form. When the HTML
Multiple-COAs, Single-Node
Plot of model: BorgsDecision

Select Observable Node(s):

<table>
<thead>
<tr>
<th>Nodename</th>
<th>Include</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fed press reports major demonstrations in Fed against war_29</td>
<td>☑</td>
</tr>
<tr>
<td>Borg believes use of strat D5 is counterproductive_26</td>
<td>☑</td>
</tr>
<tr>
<td>Borg believes use of D5 will push Fed to negotiate_30</td>
<td>☑</td>
</tr>
<tr>
<td>Borg decides to use D5 option_27</td>
<td>☑</td>
</tr>
</tbody>
</table>

Select COA(s):

<table>
<thead>
<tr>
<th>COA</th>
<th>Include</th>
</tr>
</thead>
<tbody>
<tr>
<td>dly_Med_coa_FightFirst</td>
<td>☑</td>
</tr>
<tr>
<td>dly_Fast_coa_OffensiveStrategy</td>
<td>☑</td>
</tr>
<tr>
<td>dly_Slow_coa_DefensiveStrategy</td>
<td>☑</td>
</tr>
</tbody>
</table>

Figure 7: Web page to select nodes and multiple COAs for comparison of probability profiles.

document is submitted, the probability profiles for the selected node and COAs will be displayed based on the data that was collected when the simulations were initially made. The ultimate output of the analysis of multiple COAs is a recommendation, along with the supporting rationale, for a particular COA. In rapidly evolving situations, it is typical for analysts to continually modify the influence net model as new information about the situation becomes known. With the original CAESAR II/EB tool using fixed CPN models, each time a change was made in an influence net, a new CP-net had to be automatically created and compiled before any temporal analysis could be done. This can be time consuming in a situation where speed of analysis is important. The new technique of creating a CP-net that provides a generic representation of any influence net means that changes in the influence net do not require the generation of a CP-net. The implementation of this generic CP-net and more details on the web environment are discussed in the next sections.

3 CPN Simulator for TINLs

In this section we describe the generic CPN model (Lindstrøm & Haider 2001) that can simulate any TINL when initialized with a proper marking. We remind that, in this context, a TINL consists of an influence net and timing information for the influence net. The purpose of this section is to illustrate how a simulator of TINLs can be implemented using CP-nets. Next, Sect. 4 will discuss technical issues on the architecture of the web-based environment for the TINL CPN simulator.

3.1 Motivation for the Generic CPN Model

In (Wagenhals et al. 1998) a translation from TINLs to CP-nets has been developed. The translator takes a TINL specification file as input and automatically generates a fixed CPN model which can simulate that specific TINL. It is expected that the SME wants to analyse several TINLs and COAs within a short time frame. Therefore, the time used to apply the method is of great importance. The most time consuming part of the method described in (Wagenhals et al. 1998) is that for each TINL that needs to be investigated, a complete CPN model needs to be generated with places, transitions, declarations, etc., and afterwards the simulator code should also be generated. As we will see in Sect. 5, the turn-around time for generating the CPN model, the simulator code, and then running the simulation is relatively high for non-trivial TINLs. Even though a CPN model and the corresponding simulator code of the old translation method can be generated completely automatically, it is still a relatively time consuming job to apply the method with the tools currently available.

Based on these experiences, the generic CPN model has been developed. The advantage of this CPN model is that the simulator code is generated once, and only once. In other words, we not only avoid generating a new CPN model for each TINL, but we also eliminate the need for generating simulator code for each TINL. The generic CPN model which is used to simulate any TINL, is constructed such that its behaviour is equivalent to the fixed CPN models which are generated using the old translation method. The equivalence has been investigated in (Lindstrøm & Haider 2001).

3.2 Overview of the Generic CPN Model

The generic CPN model is a hierarchical CPN model. An overview of the modules in the CPN model is given in Fig. 8. The module InitModel is used to load the TINL specification file and the other parameters into the CPN model, and will be discussed in the end of Sect. 3.5. The Top module of the CPN model gives the most abstract view. The three modules Initial, Intermediate, and Terminal model three different types of TINL nodes. Initial nodes represent actionable events, i.e. nodes without predecessors, while the terminal nodes represent the decisions nodes, i.e. nodes without successors. The intermediate nodes represent the remaining nodes with both predecessor and successor nodes.

3.3 Top Module

The net structure in the CP-net models the flow in the probability propagation algorithm, while colours are used to model the nodes, arcs and probabilities of a specific influence net. Therefore, most of the colour sets include the node identity so that it is possible to distinguish tokens belonging to different nodes even though they are located at the same place. For example, in the Top module, shown in Fig. 9, the leftmost place Trigger contains triples of node identity, the corresponding probability and initial time delay. The initial delays and probabilities for each of the nodes representing actionable (or controllable) events have been given as input in the HTML forms in Figs. 4 and 5 via the web browser.

Let us now turn to how the CPN model simulates a specific influence net. The transition Driver in the
Top module starts by removing tokens from the Trigger place and adds tokens to the place A with the initial time delays (td). That corresponds to activating the actionable events for each node at different points in time. Next, the probabilities can be propagated forward from the initial nodes via the intermediate nodes to the terminal nodes.

3.4 Intermediate Nodes

Let us consider how the nodes of a TINL are simulated by investigating intermediate nodes only. The initial and terminal nodes are modelled in a similar way and will, therefore, not be discussed here. All intermediate nodes are modelled using the same net structure. Figure 10 shows the CPN model for intermediate nodes. The model has two transitions and six places. The general idea of this module is that the transition Compute Prob models the computation of a new probability based on probabilities of the predecessor nodes, while the transition Distribute Probs models the distribution of the newly calculated probability to the successor nodes.

3.4.1 Places and Colour Sets

In this section we discuss how the static information of a TINL is modelled by describing some of the places and colour sets of the intermediate module. In the next section we will consider the modelling of the probability propagation of a TINL.

Consider the place Arcs which models the arcs in a TINL. This place always holds a single token which is a list containing an entry for each arc in the TINL. Each arc entry contains a source and a destination node id, a probability value, and a boolean control value. The control value indicates whether or not the probability of the predecessor node of the arc has been recalculated and is ready to be processed by the successor node. In other words, the control value is used to check if newly computed probabilities have been propagated forward or not. To model the arcs in Fig. 2 in Sect. 2 a list with eight entries is needed, i.e. one entry for each pair of connected nodes.

Apart from modelling the relationships between nodes, the static information of each node in the TINL must also be modelled to be able to compute the probability of each node. For that purpose, the place Rule has a token for each node in the TINL. To model a specific node, first of all, it contains the node identity of the node in the TINL so that it is possible to identify the token for a given node. The remaining values are the gate type (not discussed here), the time delay of the node from the delay file, a counter, the baseline probability, the h and g values, and a list of conditional probability weights stating how likely the node is to be true when any subset of the predecessors are true. Based on these values, it is possible to compute the post probability for a node when new probabilities arrive at the input arcs to the node.

Now the modelling of the dynamics of a TINL is discussed. When a node has propagated a new probability value forward to a successor node, the successor node must then recalculate its own probability, and then also propagate its new probability value forward to its own successor nodes, etc. This is modelled using the transitions Compute Prob. and Distribute Probs.

3.4.2 Transitions

Now the modelling of the dynamics of a TINL is discussed. When a node has propagated a new probability value forward to a successor node, the successor node must then recalculate its own probability, and then also propagate its new probability value forward to its own successor nodes, etc. This is modelled using the transitions Compute Prob. and Distribute Probs.

First consider when a node must compute its probability. It must be computed whenever one of its predecessor nodes has changed its probability. In terms of the CPN model, this means that the transition Compute Prob must occur when any of the input arcs to a given node has the control value set to one. To model this, the simulator first takes one token from each of the places Rule and Structure with the same node number (node). The variable allarcs on the arc from the place Arcs to transition Compute Prob binds to the single list on the place Arcs. Now the guard of transition Compute Prob is evaluated. First the variable input_arcs is assigned to the list of input arcs for the node by selecting those arcs from allarcs in the TINL that have endpoint in the in node. These arcs are exactly those having source in preset and destination in node. Next, the function any_recalculated tests if any of the input_arcs have the control value set to indicate that they have been recalculated by predecessor nodes.

When the transition Compute Prob occurs, a token is put on the place N with a time delay dl to indicate that the node is in the process of being updated. The probability of the node is calculated using the function compute_prob based on the probabilities of the input_arcs, and the static data on the place Rule. Finally, the control values of the input_arcs are reset using the function reset_input_controls on the arc from Compute Prob to Arcs to indicate that the probability has been propagated forward.

When the time delay has elapsed, the probability value must be forwarded to the successor nodes. In the CPN model this means that the output_arcs of the node must be updated. The transition Distribute Probs first matches a token on place N, a counter-token on place Buffer, and a token on the place Structure with the corresponding node. The guard calculates the output_arcs from list allarcs, based on the list postset for the given node. This is exactly those arcs which connect the node with its successor nodes. The function all_not_recalc ensures that all of the output_arcs have not set the control value. This is done to ensure that the probabilities have been propagated further on.

When transition Distribute Probs occurs it updates the probabilities, sets the control value of the corresponding output_arcs, and then returns the list to the place Arcs.

3.5 Initialisation of the CPN Model

The generic CPN model has to be initialised with appropriate markings to reflect the structure of a specific TINL. The data is loaded into the CPN model from the TINL-specification file, which is exported from the CAT-tool and contains all data needed to create the initial marking. The file is exactly the
same file as the one used to create the fixed model as described in (Wagenhals et al. 1998).

Figure 11 depicts the module where tokens are created and distributed to the places of the CPN model. The transition Distribute Tokens is the only transition being enabled initially. When it occurs, several things happens. First, the code segment of the transition loads the TINL-specification files for initialising the CPN model. The filename is read from the HTML form used to start the simulation. Then multi-sets of tokens are generated using functions like createBufferTokens in Fig 12. Given a list of node ids from the TINL-specification file, the recursive function generates for each node id nid, a token 1'(nid, 1). The rest of the markings for the remaining places are generated using similar functions.

The distribution of the tokens is done by means of fusion places. The places Buffer, Structure, Rule, and Arcs are fused with the corresponding places on the intermediate module in Fig. 10, and the initial and terminal modules. The place Trigger is fused with the one in Fig. 9. In this way tokens are loaded and computed in a single module of the CPN model, but are distributed to the places of several modules of the CPN model.

4 Web-Based Simulation Environment

To get an impression of the complexity of the website for the TINL CPN simulator, this section first describes the relatively complex structure of the website and describes how results can be produced via the web environment. Secondly, the actions for controlling the TINL CPN simulator are discussed. Finally, we discuss why we found using HTML forms and CGI scripts to be a good choice for this project.

4.1 Structure of Web Environment

The directory structure on the web server is depicted in Fig. 13. The topmost directories are fixed and TINL independent. The remaining directories depend on the concrete TINLs, and are created and maintained automatically by CGI scripts while the user interacts with the GUI of the web browser.
4.2 Controlling the TINL CPN Simulator from a Web Environment

In this section we describe how the TINL CPN simulator can be controlled from a web environment. We describe how the parameters for the model are retrieved from the web browser, how the simulator is initialized, and how the results from simulating the TINL are produced and shown in the web browser. The control is completely hidden from the operational planner behind the web-based interface. The technical details of the method which is used for controlling the CPN simulator from a web browser is described in detail in (Lindstrøm 2001).

The simulator is controlled automatically by a batch script when the CGI script is executed. The source code for the batch script is not included here. Instead, we will discuss the elements of the batch script at a conceptual level. Below we first show the sequence of actions in the batch script used for controlling the TINL CPN simulator when the CGI script is invoked from an HTML form. Afterwards, we discuss more details on the actions of the batch script.

1. Retrieve the model name and delay name from the HTML form.
2. Load a TINL-specification file and a delay file.
3. Retrieve the TINL specific parameters from the HTML form.
4. Create output directories.
5. Save an HTML document with input parameters.
6. Calculate initial marking.
7. Run simulation and collect data.
8. Generate and save graphics using gnuplot.
9. Produce and save HTML documents containing results for later use.
10. Send the summary HTML document to the web browser.
11. Quit the CGI script.

When the CGI script is invoked, the batch script retrieves the parameters from the HTML form. However, the parameters are dependent on the given TINL. Therefore, it first retrieves the model name and the delay name to know in which directory the TINL specification file and the delay file can be found. Based on the information in the TINL specification file, the remaining parameters can be retrieved from the HTML form.

The next action is to prepare for producing output while simulating the model. Therefore the necessary output directories are created, and an HTML document (run-cpn.html) equal to the one used for setting parameters for this simulation – but including the values typed in by the operational planner – is saved to store all initial parameters for this specific delay file and COA. This document can be used to run a similar COA with a few changed parameters. This makes it easy to specify several similar COAs.

The parameters of the simulator are then initialised with the values from the TINL specification file, the delay file, and the HTML form. Afterwards, the simulator is ready to be started. While simulating, data will be collected and stored in the output directory.

After the simulation has stopped, the data which has been collected during the simulation is used for generating graphs. The graphs are created by first generating a file with a script for the graph plotting tool gnuplot (Gnuplot 2002), and then gnuplot is executed with the script as parameter. A graph (res<nodeno>.png) is created for each of the nodes which the operational planner has marked as observable in the input HTML form. These graphs are used to give a very detailed profile of each individual node. Another graph (comp.png) which contains graphs from all observable nodes is also generated. It is used to compare the profiles of the observable nodes with each other. An example of such a graph was given in Fig. 6 in Sect. 2.

Next, the file detailedindex.html containing an HTML document including the graphs of the individual observable nodes, is generated and saved. Finally, an HTML document (index.html) is generated to give an overview and access to all the HTML documents and simulation results. This document is saved in a file, but is also sent to the web browser to give the SME immediately access to an overview of the simulation results.

4.3 Why Choosing a Web-Based Simulation Environment

A web environment is useful for this project due to the fact that the SMEs can easily access the simulation application from anywhere. From the modeller’s point of view it is also easy to create a domain-specific GUI using standard web techniques.

Using HTML forms and CGI scripts for controlling the simulation environment has turned out to be a good choice for this application. First of all, because it requires only a few basic programs like the interpreter Perl (Wall et al. 1996) and the Standard ML runtime environment to be installed on the web server. Secondly, the CGI script containing the TINL CPN simulator code is generated completely automatically from Design/CPN, and is only generated once when installing the web-site.

The most notable disadvantage is that the TINL CPN simulator has a size of about 12 MB, and the memory image which takes up the memory while executing the CGI script is also rather large. However, one of the assumptions of the web environment is that only a few SMEs are using the CGI scripts at the same time. Therefore, the relatively large memory requirement of the web server is not a critical problem. If it turns out to be a problem in practice, then the TINL CPN simulator of the CGI scripts may be distributed to other machines.

5 Performance Results on Execution Time

In this section we present results on the time-effectivity of the developed TINL CPN simulators. As mentioned previously, the time it takes to apply the method is relevant because the generic CPN model has been developed to be able to do faster analysis of TINLs than previous CPN models. We compare the effectivity of the generic CPN model presented in Sect. 3 with the fixed CPN models obtained using the method presented in (Wagenhals et al. 1998). In addition, the generic CPN model has been simulated using both the old simulator and the new simulator (Mortensen 2001) of Design/CPN. The reason for using the new simulator of Design/CPN is that it turned out that the old simulator was too slow for the folded CPN model. Table 1 contains execution time in milliseconds for two TINLs with 5 and 30 nodes. A 30 node TINL can be considered as a TINL with a real-world size.

For the fixed CPN models there are extra time penalties for drawing the actual CPN model which are not present for the generic CPN model. The table shows that it takes more than 20 seconds to draw the full CPN model. In addition, extra time is needed for generating simulation code (switch) for the CPN model. The switch time depends heavily on the number of nodes in the TINL. The 5-node TINL takes about 86 seconds to switch, while the 30-node TINL takes about 396 seconds. The simulation time for the fixed models is very low. The reason is that the old simulator code is quite effective for CP-nets with few tokens on each place, which is the case for the fixed CPN models. The overall time it takes to draw, switch, and simulate the CP-net of a TINL using the fixed CPN model is relatively high: more than 100 seconds for a small TINL with only 5 nodes.

The generic CPN model avoids the need of drawing and switching a CPN model each time a new TINL is to be analysed. The switch is made only once for the generic model, and the simulator can be applied to any TINL afterwards. Therefore, the switching time is not included in the table. Simulating the generic CPN model using the old simulator is comparable to simulating the fixed model for very small TINLs. However, as the simulation time of the 30-node TINL shows, it takes longer time to simulate the generic CPN model for TINLs with 30 nodes. The reason is that the old simulator is ineffective when simulating CPN models with many tokens on a place. When considering the total time, the generic CPN model using the old simulator is almost as inefficient as using the fixed CPN model.

The large simulation time of the generic CPN model has been solved using the new CPN simulator which is optimised for simulations with many tokens on a place. The improvement can be seen from the fact that the simulation time of the generic CPN model using the new simulator is close to the simulation time of the fixed CPN model using the old simulator. The simulation time for the generic CPN

\[ \text{We have used a Sun 4, 128 MB RAM, sparc-solaris 5.8.} \]
<table>
<thead>
<tr>
<th>#TINL Nodes</th>
<th>Model</th>
<th>Simulator</th>
<th>Draw CP-net</th>
<th>Switch</th>
<th>Simulate</th>
<th>Total Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Fixed</td>
<td>Old</td>
<td>21,390</td>
<td>86,000</td>
<td>100</td>
<td>107,490</td>
</tr>
<tr>
<td></td>
<td>Generic</td>
<td>Old</td>
<td>-</td>
<td>-</td>
<td>140</td>
<td>140</td>
</tr>
<tr>
<td></td>
<td>Generic</td>
<td>New</td>
<td>-</td>
<td>-</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>30</td>
<td>Fixed</td>
<td>Old</td>
<td>24,860</td>
<td>396,000</td>
<td>390</td>
<td>421,250</td>
</tr>
<tr>
<td></td>
<td>Generic</td>
<td>Old</td>
<td>-</td>
<td>-</td>
<td>320,440</td>
<td>320,440</td>
</tr>
<tr>
<td></td>
<td>Generic</td>
<td>New</td>
<td>-</td>
<td>-</td>
<td>520</td>
<td>520</td>
</tr>
</tbody>
</table>

Table 1: Execution time in milliseconds for different models using different simulators.

model includes the first step where the data is loaded into the model and tokens are generated. However, for the 30-node TINL specification file the first step takes only about 50 milliseconds. In conclusion we note that the simulation time using the new simulator and the generic CPN model is fully acceptable for web simulations.

6 Conclusion

In this paper we have presented a web-site which contains a CPN simulator for effect based operational planning. The CPN model is used to evaluate alternative COAs with respect to timing and probability profiles. The input for the model consists of a complete influence net, a timing profile for actionable events, and estimated probabilities for the events in the influence net. The results of simulating the CPN model are, e.g. graphs displaying expected probabilities for each of the observable nodes at different points in time. This information can be used in the process of timing an operation by selecting the best of the examined COAs.

Future work may include embedding the TINL CPN simulator into the CAT tool as a component. The simulation results could then be displayed within the CAT tool, and thereby lead to a more iterative evolution of TINLs. It would probably be more effective to apply the method if, when adding an extra node to a TINL, one could see the consequences on the simulation results immediately after adding the node.

The IEEE 1516 standard called High Level Architecture (HLA) (HLA 2002) can be used for interconnecting simulation models to become so-called federations of simulation models. By creating an interface for the HLA runtime infrastructure to the TINL CPN simulator, it would be possible to use output from simulations of e.g. a war game as input to the simulator. There exist simulation models at different strategic levels of war planning, and the outcome of using the TINL simulator in such a federation could be a method to easily determine the effect of alternative COAs based on these other simulation models.

Acknowledgements. This research was conducted in cooperation between University of Aarhus and the C3I Center of George Mason University, with partial support provided by the U.S. Office of Naval Research under Grant No. N00014-01-1-0538.

References


HLA (2002), HLA Standards Development.


