# Role-Based Extended Petri Net Models and their Applications

Fred D.J. Bowden, Mike Davies and John M. Dunn

Information Technology Division
Defence Science and Technology Organisation, Salisbury
PO Box 1500,
Salisbury, South Australia 5108

**Abstract** Extended Petri nets (PN) have been used to simulate many different systems including computer hardware and software, assembly lines, communication networks and more recently command, control, communications and intelligence (C3I) systems. Extended PN are a powerful tool in the modelling of C3I systems as their structure gives them the inherent properties of concurrence and hierarchy. It is these features that make them ideal for the modelling of decision making and other processes in a C3I system. In this paper a role-based methodology for the design of entities in a C3I system is described, where roles are considered to be functional elements of the given entity and a role itself can be made up of a number of roles. The hierarchical qualities of this approach allows for both top-down and bottom-up design. Given the design methodology, extended PN are the ideal tool for this type of modelling. One drawback of PN is their complexity to someone unfamiliar with their representation and structure. To overcome this problem an explanation and analysis capability has been developed in the declarative language Prolog, this capability helps convey the underlying assumptions and allows the designer to relate the PN model characteristics in a clear form. This paper presents the role-based methodology and the explanation and analysis ability; both of these are general-purpose and applicable to any PN representation.

## 1. INTRODUCTION

An important element in many organisations is effective command, control, communication and intelligence (C3I). To assist the Australian Defence Organisation, Information Technology Division of the Defence Science and Technology Organisation is developing computer software to enable the study of C3I systems effectiveness. The nucleus of the software is known as the Distributed Interactive C3I Effectiveness (DICE) simulation (Davies, Gabrisch [1995]).

The DICE simulation represents the C3I system it is modelling as a number of interacting nodes. Each node represents a different entity in the real C3I system and its external environment. The links between the nodes are communications between the respective entities in the real system. Communication in DICE is carried out by the passing of formatted textual messages. There are three different types of nodes in the DICE simulation: interactive players, peripheral units and artificial agents.

Interactive players in the DICE simulation will generally be used to represent decision makers or commanders in the C3I system being studied. Interfaces allow the players to interact with the other nodes in the DICE simulation regardless of whether they are interactive or non-interactive nodes.

In some cases there may be existing models and simulations that represent a given aspect of the system being studied, for example an aircraft simulation may be used to calculate the position of aircraft. When models and simulations are incorporated into a given DICE scenario they are referred to as peripheral units. In general, peripheral units do not model C3I aspects of the system but are used to assist in measuring the effectiveness of the C3I system by helping form a representation of the overall system and its mission or objectives (Davies, Gabrisch [1995]).

As is frequently the case with interactive simulations, there is often a need to use some form of non-interactive nodes to represent different aspects of the system (Lewis [1994]; Lankester et. al. [1994]; Laird et. al. [1994]; Cox et. al. [1994]). In DICE each time a node cannot be represented by a peripheral unit or does not have a human player representing it, an artificial agent is designed to perform its task. The artificial agents in DICE are extended Petri net (PN) simulations and are considered to be made up of a number of roles which are brought together to form the overall artificial agent structure. Each role may also be constructed from a number of roles, thus forming a hierarchical structure where the detail increases with depth. This document considers how these role-based artificial agents are designed and incorporated into the DICE simulation.

It should be noted that there will be times when studies will be carried out using a non-interactive simulation. In these cases the DICE simulation will be used and all the nodes will be either represented by peripheral units or artificial agents.

## 2. ARTIFICIAL AGENT REQUIREMENTS

In the DICE simulation, it is desirable that the artificial agents must communicate with the real players in a way which will make interactive players unable to determine whether they are communicating with a real or artificial decision maker. The need for seamless integration is internationally recognised (Lewis [1994]; Cox et. al [1994]) and is very important to maintain a realistic simulation of the real environment. This is particularly true in training exercises, in that interactive players cannot be allowed to determine between which nodes are artificial and which are real as this knowledge may prejudice how they treat the other nodes. A certain degree of seamless interaction is also desirable in analytical simulations. To address this problem in the DICE simulation, a standard language based on formatted textual messages has been chosen for communication between nodes (Davies, Gabrisch [1995]).

### 2.1 Basic Artificial Agent Structure

Initially, the artificial agents in DICE will be rule-based; more sophisticated representations may be adopted as the field of artificial intelligence is researched and practical benefits determined. It has been expressed that current machine learning techniques are not sophisticated enough to deal with a problem this complicated (Cox et. al. [1994]).

The fact that the artificial agents are rule-based, leads to there being a natural break down of their processes into smaller components. These components will be loosely termed "roles". A role is not necessarily the base element of the artificial agent (as in the work by Levis [1993]) it is more a function, which may in turn have many roles within it. This structure leads to a hierarchical design technique. By taking a hierarchical approach (Aronson [1994]) the complexity of the design can be determined by the work being carried out. This method also supports the natural break down of complex systems into simpler ones. This means the roles can be designed initially and then brought together to form the overall artificial agent, leading to a bottom up design approach. Alternatively the designer can initially sketch out the basic functions of the artificial agent and then add detail by enhancing its roles independently, leading to a top down modelling approach. The artificial agent being modelled and the information about the artificial agent will determine which method is most appropriate for the task at hand, making it necessary to allow either approach. This method also simplifies modifications to the artificial agent since only appropriate roles need to be changed not the agent overall.

C3I networks involve many different systems working both concurrently and consecutively towards the same main objective. This is also true for each of the nodes in the system. Thus, an artificial agent may involve events which occur either in series or in parallel. This is an important feature of any decision making process and so must be reflected in the artificial agent design. To some extent this is dealt with by taking the role approach, where roles are arranged according to the way their related functions occur in the real system.

### 2.2 Explanation and Analysis Capability

Adequate credibility and realism in artificial agents is essential; judgement of such qualities can generally be best made by the experts in the real systems being modelled. The underlying assumptions and characteristics of artificial agents need to be conveyable in a form easily understandable to such an expert. This expert should be able to interrogate features of the artificial agents and to form some judgement on their realism. Having an expert's endorsement of the assumptions used in such representations is a vital prerequisite to any C3I system study. The ability to have artificial agents, for example, explain their actions in an easily understandable form, is recognised as a very important requirement (Cox et. al. [1994]; Lewis [1994]).

## 3. THE USE OF EXTENDED PETRI NETS TO REPRESENT ARTIFICIAL AGENTS

Having defined the requirements of the artificial agent design in the DICE simulation it was then necessary to find an efficient method of modelling artificial agents with these features. Many different methods were considered (Bowden et. al. [1995]), before the decision on the use of extended Petri nets (PN) was made.

PN have been used in the study of C3I systems by many researchers and even applied to the modelling of decision makers (Levis [1993]). As such they have not been applied to the design of artificial agents in a distributed interactive simulation environment in the way required in DICE. However, PN have all the required features to model the operation of the DICE artificial agents.

PN have a graphical modular representation that allows the artificial agent designer to represent the role logic in a straightforward manner. Each role can be thought of as a PN which is then joined to other PN via common places and transitions, this allows each of the roles to be designed individually and then brought together. Alternatively a PN can be constructed outlining the artificial agent structure and then refined by adding detail. Thus artificial agents can be designed in the same fashion as is described in section 2.1.

Due to the PN representation, they are easy to modify and the effect of changes can be clearly seen. An inherent feature of the PN structure is that it allows for the modelling of features such as synchronisation, concurrence and resource sharing, which are features required of artificial agents in DICE. In the early development of PN they had the drawback of becoming very complicated and confusing when the complexity of the system increases. However, PN have been extended by adding elements such as coloured tokens and hierarchies which reduce the complexity of their graphical representation for large and/or complex systems.

In representing an agent the aim is to model a system that comprises discrete events occurring under defined conditions. It is this that makes extended PN the ideal tool for modelling such systems, as this is the type of systems PN were designed to model. It should be noted that many similar methods can be shown to be either equivalent to or subclasses of PN (Peterson [1977]).

## 4. SIMPLE EXAMPLE OF ROLE-BASED AGENT

The following sections introduce a simple example of a role-based agent and refer to this example to illustrate many of the concepts associated with the role-based methodology and other features. Artificial agent design and implementation will be achieved through use of the Petri net analysis environment that is currently being developed, which will automate many of the procedures that are discussed below. It should be noted that the example given here is purely being used as a way of demonstrating the role-based architecture described above and the reader should not be concerned about the mechanics of the PN.

### 4.1 Designing Roles

The role-based architecture of the DICE artificial agents is illustrated in the example given in Figure 1. The notation used for the PN in this example is that defined by Jensen [1990].

Consider the situation where a pilot requests permission from an air traffic control tower to taxi to a runway so that he/she can begin preparation for take off. If permission is given then the pilot will proceed along the taxiway, otherwise he waits for a period of time before submitting the request again. If a plane is taxying, then the taxiway is considered unavailable to other aircraft. On receiving a request the control tower must check the current weather conditions and then the availability of the taxiway before deciding if the pilot is cleared to taxi.

Figure 1 shows the role-based model representation of this system. In Figure 1 (a) the higher level model defining the roles as seen at the pilot level is shown. First he submits his request, which is represented in the PN by the firing of transition *t1* and the creation of an *r* token in place p2. Next the control tower responds, either approving or disapproving the request, ie transition *T1* fires. If the pilot is given permission to taxi then a *p* token is created in *p3*, otherwise an *r* token is created in *p1*. If the request is approved the pilot proceeds to taxi, transition *t2* fires on completion of the taxying placing an *f* token in *p4* to indicate the pilot has completed his taxi and a *t* token in *p5* signifying the availability of the taxiway.

In Figure 1 (b) the details of the control tower are modelled. The extended PN shown here is a substitution transition to transition *T1* in the PN of Figure 1 (a). This model involves two different roles; the checking of the weather conditions (transition *ta*) and taxiway (transition *tb*). Also given in Figure 1 is a description of the physical meaning associated with the tokens of these nets. To make the associations between the two nets easily seen, like places and tokens have been given the same names.

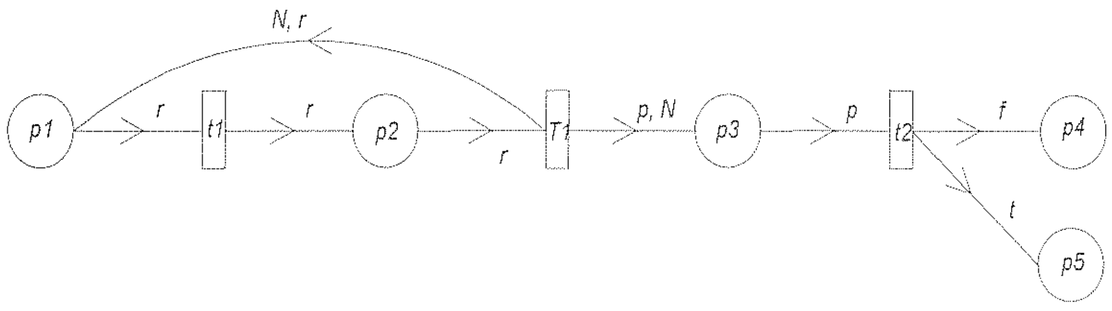
The above example shows a top down approach to modelling. Initially the basic top level structure of the pilot net was defined and then the control tower role was modelled in more detail. An alternative bottom up design can be illustrated by considering the design of the control tower.

Figure 2 shows the two roles that exist in the control tower. Figure 2 (a) shows a general "checking of conditions" role and Figure 2 (b) shows a general "checking the availability of a resource" role. Having designed these two roles we can then bring them together to form the control tower modelled in Figure 1 (b). The two roles in Figure 2 could be regarded as existing in a library of roles from which they are selected and employed in constructing the control tower role. It is important to note that the lower level roles are originally general but become instantiated with a particular purpose when employed in the control tower role. The condition to be checked becomes the weather and the resource being checked for availability becomes the taxiway. The Petri net analysis environment will allow for any combination of the above role-based design approaches as well as giving the designed systems the analysis and explanation capability required of them in the DICE project.

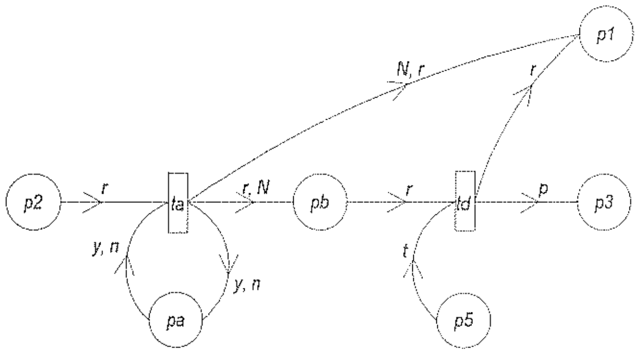
It should be noted that the control tower net represented here only makes up a very small part of what would be a complete control tower model. The portion shown is that activated by the particular problem being considered. This illustrates another nice feature of the PN role approach to agent design, only the relevant part of the PN model is activated when called upon.

### 4.2 Explanation and Analysis Capability

A PN explanation and analysis capability has been established using the declarative language Prolog. Prolog is a particularly suitable language for describing PN, where the basic connectivity characteristics can be regarded as a set of facts or clauses plus logical conditions. Perhaps the most powerful feature of Prolog, with regard to establishing an explanation and analysis capability, is its search and multiple solution identification abilities. The analysis component interrogates a given PN according to user-specified goals or



(a) Pilot Net



(b) Control Tower Net

| Token | Description |
|---|---|
| r | Signifies the submission of a request to taxi. |
| p | Signifies the control tower giving permission to taxi. |
| f | This token in place p4 indicates the pilot has completed taxying. |
| t | This token in place p5 implies that the taxiway is available. |
| y | This token in place pa implies the weather condition is satisfactory. |
| n | This token in pa implies the weather condition is unsatisfactory. |
| N | This token represents the empty set, it implies that no token is passed. |

Figure 1: Aircraft Ground Control Example of Role-Based Agent

queries whilst the explanation capability abstracts and translates the query results such that they are presented in a form more easily understood to someone less versed in PN notation. The explanation capability is achieved through the use of declarative tags on the PN that give descriptions of the significance of a particular token residing at a particular place in the net; the firing of a particular transition mode; and summary information for an instantiated role.

A number of important queries were determined and are discussed below. The Prolog software automatically identifies any multiple, ie alternative, solutions to any query. The presence of roles allows explanations to be presented at a level appropriate to a user's query. A graphical user interface environment associated with this capability has been developed.

### 4.2.1 Reaction to a given situation

In a PN, a situation is described by a certain marking which, in raw Prolog form, might be [[p2,r,1],[pa,n,1]]. The PN explanation capability allows this situation to be expressed in the form:

['Permission to taxi requested',
'ct: Weather not suitable for taxi']

and it is in this form that a user can specify a situation. Consider specifying this situation and indicating that the level of abstraction required (the *target net*) is at the Pilot net. A query for the reaction to this situation would return an outcome of:

['Control Tower determined that permission should not be granted for taxi'].

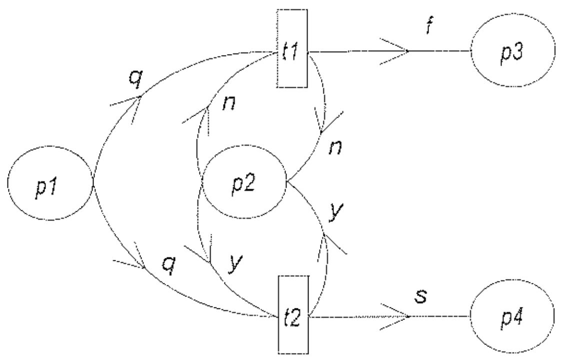
The mode that has actually fired in response to this goal is one associated with the Check Conditions net which checked the conditions (the weather) and returned in the negative. However, abstraction was used through the Control Tower net to the Pilot net resulting in the summary information above.

### 4.2.2 Sequence of Actions Resulting From a Given Situation

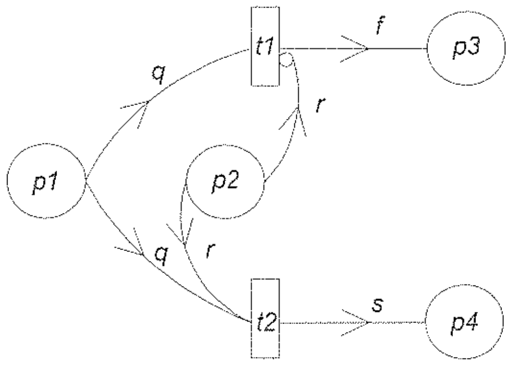
This clause generates an outcome consisting of a sequence of actions that occurs in response to a stated initial situation and resulting in some fully or partially defined final situation.

For example, with a target net of Pilot, if the initial situation were:

[' Permission to taxi requested',



(a) Check Conditions Role



(b) Check Resource Availability

| Token | Description |
|---|---|
| q | Represents the submission of a query on a condition or the availability of a resource. |
| f | Indicates conditions are not satisfactory or the resource is not available. |
| s | Indicates conditions are satisfactory or the resource is available. |
| y | Corresponds to satisfactory conditions. |
| n | Corresponds to unsatisfactory conditions. |
| r | Represents the resource, its presence in p2 indicates that the taxiway is available. |

Figure 2: Control Tower Roles

'ct: Weather suitable for taxi',
'ct: Taxiway available']

and the final situation were any situation which contains the state:

['At runway']

then the response to this query would be the outcome:

[['Control Tower determined that permission should be granted for taxi'],
['Taxied to runway']].

Changing the target net to the Control Tower level would result in the outcome:

[['ct: Determined that weather is OK for taxi'],
['ct: Determined that taxiway is available for taxi']]

which indicates the same outcome but in terms relevant to the Ground Control net.

### 4.2.3 Explanation of Actions

This query gives the ability to select a particular action (or group of actions) and ask the question "Why did/would this action occur?". In the case of a role-based PN, actions will either be associated with the firing of regular transition modes or effective firings of substitution transitions. An explanation of why an action occurred or would occur is given, then, by the input requirements of the associated mode or a description of the mode summarised by a substitution transition.

### 4.2.4 Sensitivity of Actions That Could Arise From a Given Situation

This query refers to taking a specified situation and requesting an illustration of the sensitivity of any actions (ie mode firings) to this situation. Processing of this query involves determining which firings are dependent on part or all of the given situation and ascertaining to what extent that situation would need to change in order for that firing to occur.

### 4.2.5 Actions That Could Result in a Given Situation

This query determines what groups of modes can fire such that their combined firings result exactly in a given situation. It is important to note that firing time considerations are not made here; the clause will return modes whose outputs upon firing cause a given situation, the modes do not necessarily fire simultaneously.

### 4.2.6 Actions That Could Contribute to a Given Situation

This query determines what actions could contribute to a given situation, ie what firings have output that is sensitive to the situation concerned. The result of any firings might not be exactly the required situation but could contribute to it. This approach is particularly important when multiple or repeated firings of a mode are required in order to produce a given situation.

## 5. CONCLUSIONS

This paper has outlined a role-based methodology for designing artificial agents in the DICE simulation. This methodology allows for both top down and bottom up design, and uses extended PN to describe the roles. It should be noted, this approach is not limited to artificial agents, it could be extended to the modelling and simulation of the complete system.

An explanation and analysis capability has been developed to accompany the PN technique. However, the explanation capability is only as good as the quality of the declarative tags attached to the PN and this will be the subject of further research as is addressing the transient and stochastic features that an agent can possess. Research is also being carried out to develop a PN extension designed exclusively to deal with the problems associated with role-based agent modelling.

## 6. REFERENCES

Aronson, J., The SimCore tactics representation and specification language, paper presented at 4th Computer Generated Forces and Behavioral Representation Conference, Orlando, Florida, 1994.

Bowden, F.D.J, Davies M., Dunn, J.M., Representing role-based agents using coloured Petri nets, paper presented at 5th Computer Generated Forces and Behavioral Representation Conference, Orlando, Florida, 1995.

Cox, A., Gibb, A., Page, I., Army training and CGFs - A UK perspective, paper presented at 4th Computer Generated Forces and Behavioral Representation Conference, Orlando, Florida, 1994.

Davies, M., Gabrisch, C. The distributed interactive C3I effectiveness (DICE) simulation project: an overview, paper presented at 5th Computer Generated Forces and Behavioral Representation Conference, Orlando, Florida, 1995.

Jensen, K., Coloured Petri nets: A high level language for system design and analysis, *LNCS 483*, Advances in Petri Nets, 342-416, Springer-Verlag, 1990.

Laird, J.E., Jones, R.M., Nielsen, P.E., Coordinated behavior of computer generated forces in Tac-Air-Soar, paper presented at 4th Computer Generated Forces and Behavioral Representation Conference, Orlando, Florida, 1994.

Lankester, H., Robinson, P., GeKnoFlexe a generic, flexible model for C3I, paper presented at 4th Computer Generated Forces and Behavioral Representation Conference, Orlando, Florida, 1994.

Levis, A.H., Colored Petri net model of command and control nodes, Toward a Science of Command, Control, and Communications, *Progress in Astronautics and Aeronautics*, Volume 156, Editor Carl Jones, 181-192, 1993.

Lewis, J.W., Agents that explain their own actions, paper presented at 4th Computer Generated Forces and Behavioral Representation Conference, Orlando, Florida, 1994.

Peterson, J.L., Petri nets, *Computing Surveys*, Vol. 9, No. 3, 223-252, 1977.