

Applying Coloured Petri Nets and Design/CPN to an Air-to-Air Missile Simulator

Steven Gordon and Jonathan Billington

School of Physics and Electronic Systems Engineering,
University of South Australia, Adelaide, SA 5095, Australia
Email: [sgordon, jb]@spri.levels.unisa.edu.au

Abstract

In this paper the communication mechanisms of a missile engagement simulator are modelled and analysed. The simulator is developed as a testing platform for missile guidance and control algorithms. The simulation uses concurrency and remote execution concepts to enhance performance. Coloured Petri nets are a well suited formal approach for modelling and analysis of these concepts. Design/CPN is used to create and analyse the model of the simulation. A new requirement of the graphical user interface is identified for the simulation to operate successfully. The communication mechanisms are without deadlocks and are suitable for the simulator.

Keywords: Missile Simulator Design, Distributed Systems, Coloured Petri Nets, Formal Analysis

1 Introduction

Computer simulations of a missile engaging its target provide an environment for testing the guidance and control functions of the missile. The accuracy of the tests depends on the detail of the models used (eg. target, missile, etc.) and correct communication between the models. This paper addresses the problem of analysing the communication mechanisms for an Integrated Weapons Simulator (IWS) [CGW97a].

IWS is an Air-to-Air missile engagement simulator with a graphical user interface (GUI). Its development, from requirements specification through to testing and delivery, was part of a final year undergraduate project in Computer Systems Engineering at the University of South Australia. The three project members had joint supervision from staff of University of South Australia and the Defence Science and Technology Organisation (DSTO) Australia.

DSTO Australia, the research arm of Australia's Department of Defence, will use IWS to test various algorithms for the guidance and control of Air-to-Air missiles. To provide accurate simulations, the complexity of these algorithms may be large. Therefore, in designing IWS, two important features are desired so that the performance of the

system is adequate: concurrent execution and remote execution of separate components of the simulation. In providing these features it is necessary to verify that communication between the components is correct. Coloured Petri Nets (CP-nets) [Jen92] are well suited to modelling concurrency and hence are a suitable formal method for this problem. CP-nets have been successfully used to model and simulate concurrency in many applications [Jen97], and also for applications that use concepts of distributed execution [JM96].

In this paper CP-nets are used to model the communication mechanisms in IWS and to analyse them for deadlocks. An overview of the operation of IWS and its components is given. Then the CP-net models of IWS are described. An analysis of the entire simulation model is made, and finally the conclusions drawn from the analysis are presented.

2 The Missile Engagement Simulation

2.1 Structure and Function of IWS

IWS is developed to give DSTO Australia a testing platform for missile guidance and control algorithms. Hence, IWS is required to provide appropriate testing functionality and mechanisms for easily changing the algorithms. The first requirement is implemented by the GUI. The second requirement is met by structuring IWS into components that represent the relevant functions of a missile engaging a target. Figure 1 shows the physical system IWS models.

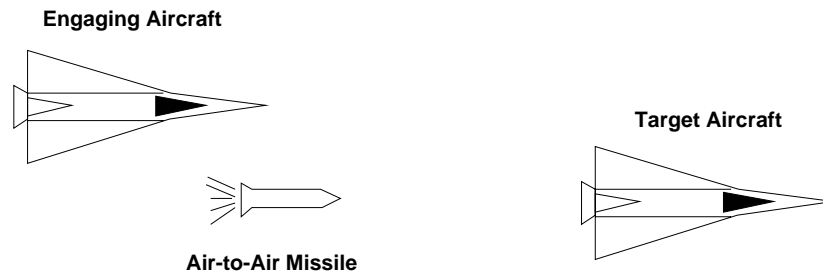


Figure 1: Physical System

When the engaging aircraft detects the target, it launches an Air-to-Air missile. After launching, the missile uses its own guidance system to track the target. IWS begins simulation from this stage - it does not simulate the launching aircraft or procedure.

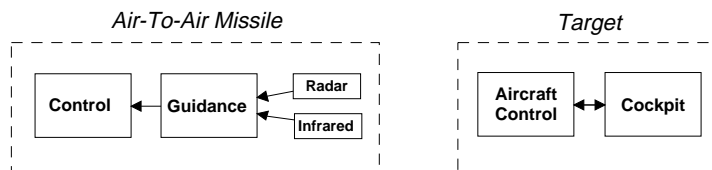


Figure 2: Missile and Target Physical Structure

The physical components of the missile and target that IWS is concerned with are shown in figure 2. The Air-to-Air missile has two detection mechanisms - a Radar (RF) and an Infrared (IR) sensor. These are physical devices on the missile that detect the location of a target. Both mechanisms are used to improve accuracy. For example, the

Radar may give inaccurate data if electronic counter measures are taken by the target. In this case, the Infrared data will be used. The data to be used is determined by the fusion mechanism in the Guidance component of the missile. The Guidance component then calculates the trajectory of the missile required to intercept the target. The Control algorithm controls the thrusters on the missile for it to maintain the calculated trajectory.

To simulate this physical system, IWS is divided into five components: Target, Radar, Infrared, Missile Control, and GUI. The components, and the interaction with the user, are shown in figure 3.

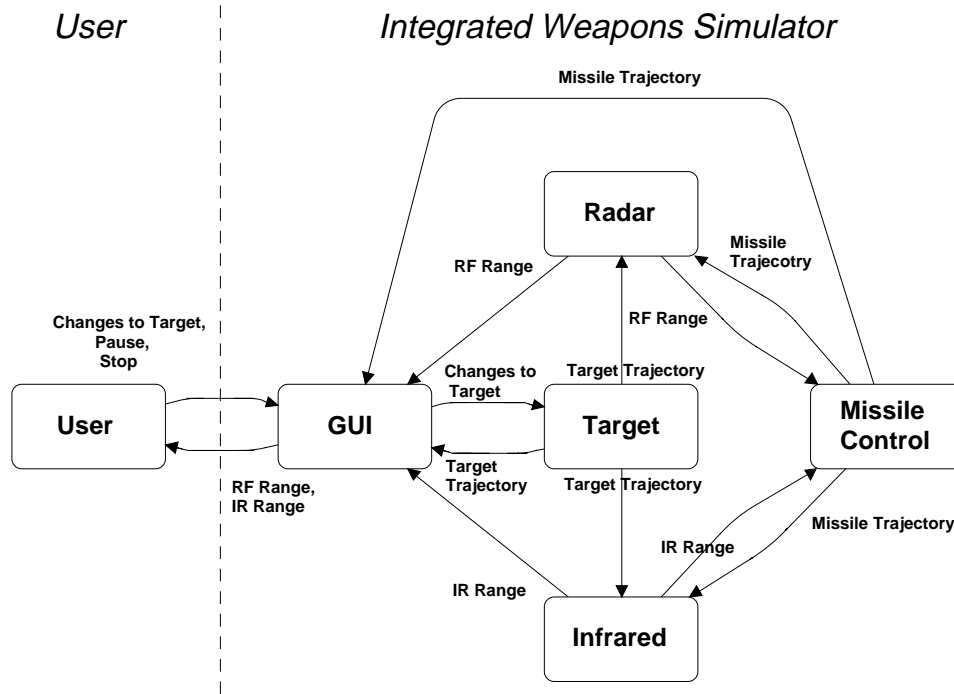


Figure 3: IWS Components

- **User.** The user of IWS plays three roles during the simulation:
 1. running the simulation, eg. issuing “pause” and “stop” commands,
 2. controlling the target by giving required changes to the target’s trajectory, and
 3. viewing the target from the missile’s point of view, ie. using the radar and infrared ranges.

To do this, the user interacts with the GUI. (Future versions of IWS could separate the roles by introducing a new user who controls the target, while the first user runs the simulation from the missile’s point of view [CGW97b].)

- **GUI.** The GUI component accepts inputs from, and presents the results of the simulation to, the user. Data from the Radar and Infrared systems are shown, and continuously updated, on a display for the missile. Inputs to Target are entered via a target controller window. This window, which allows changes in speed, elevation and azimuth, represents the throttle and control stick of the target aircraft.

The GUI also starts and stops the other four simulation components. It begins by sending the user inputs (changes to target trajectory) to the Target component. Once results from Target and Missile Control are received, the GUI compares the target and missile positions. If the two are the same, ie. the missile has hit the target, the simulation stops. Otherwise, the next user input is sent to Target and the simulation continues.

- **Target.** The Target component models a target for the simulation. The model must be able to perform realistic maneuvers so the guidance and control functions of the missile can be fully tested. A trajectory is calculated by Target based on inputs from the user (via the GUI). The inputs indicate a change in speed and direction. With these changes, the new position and velocity of the target are sent to the GUI, Radar and Infrared. Note that only a single target is modelled - IWS does not consider multiple targets [CGW97a].
- **Radar.** The Radar component simulates the physical radar sensor on the missile. It requires both the missile and target current positions and velocities. The Radar calculates the range (distance, elevation, and azimuth) of the target from the missile. To model the inaccuracies of the radar, small amounts of noise are added to these calculations. The range is sent to Missile Control.
- **Infrared.** The Infrared component simulates the physical infrared sensor on the missile. It requires the same inputs as the Radar and sends a calculated range to Missile Control. Again, the inaccuracies of the infrared sensor (due to, for example, cloud coverage) are modelled by adding noise to the calculations.
- **Missile Control.** The Missile Control component includes the guidance and control functions of the missile. The ranges from the Infrared and Radar systems are used to calculate the required missile trajectory. In IWS, the magnitude of the missile velocity (the speed) is constant. Once these guidance and control functions are performed, Missile Control then simulates the actual motion of the missile. The resulting position and velocity are sent to the GUI, Radar and Infrared. Once successfully tested, it is proposed that the guidance and control algorithms used in Missile Control can be included in the software onboard an Air-to-Air missile.

2.2 Communication Mechanisms

To achieve a likeness to the real-life situation, IWS executes each component concurrently. However, as the inputs of some components depend on the output of others, they cannot always be executing at the same time. For example, although in the real world the target and missile are independent, in IWS parts of the missile (Radar and Infrared) are synchronised with the target. There are several advantages of this mode of execution [SG94], the main one being computation speedup. This is particularly useful when the GUI must update its display (a CPU intensive task) and the other components can continue. The concurrent components are known as *threads* (eg. Target thread, Radar thread, etc.).

In addition to concurrency, the four simulation components (Target, Radar, Infrared, and Missile Control) will have the ability to be executed remotely. This will allow the resources for IWS (for the simulation) to be distributed across multiple computers [Tan92]. The section of each component executing remotely is called a *process*. The process will perform the calculations for models used in each simulation component.

Using both threads and processes in IWS introduces three communication mechanisms.

1. *Thread-to-thread* - Each of the five threads use shared memory to communicate, with access guarded by semaphores. This is a simple and common technique for multi-thread communication. The threads are required to run on the same computer.
2. *Thread-to-process* - The four simulation threads communicate with their corresponding processes using TCP/IP sockets. TCP/IP is the most common set of protocols used for communication over the Internet [Los95]. TCP/IP sockets are a standard feature of UNIX and as IWS will be developed for a UNIX environment [CGW97a], this method of interprocess communication allows IWS to run on almost any UNIX-based computer.
3. *Process-to-process* - Direct communication is used between the Radar and Missile Control processes and between the Infrared and Missile Control processes. The ranges are sent directly to Missile Control because the GUI is not required to control the missile components (Radar, Infrared and Missile Control). However, there is a need for the GUI to control the Target and the missile components separately, hence there is no direct communication between the Target and Radar or Infrared.

For successful operation of IWS, it is required that each of the above mechanisms are tested to operate correctly. This includes analysis to determine whether the flow of data is correct and that the terminal state (the state the system is expected to stop in) is always reached.

3 CP-nets

CP-nets [Jen92, Jen94] are a class of high level nets that extend the features of basic Petri nets. The two most basic and important differences are that CP-nets use: tokens which are arbitrarily complex data, and inscriptions on arcs and transitions. With the use of hierarchies, these features make CP-nets a powerful modelling tool for large applications.

The CP-nets in this paper were edited, simulated and analysed using Design/CPN [Sof93]. The Occurrence Graph (OG) tool [CJ95] of Design/CPN was used for analysis. This tool allows the creation of the full occurrence (or reachability) graph and also provides the capability to query the OG to determine properties of the system being modelled.

The choice of CP-nets as a formal method for modelling and analysis of IWS was based on: the existing experience with CP-nets, and their natural capability of modelling concurrent events.

The CP-nets were created by one member of the IWS Project group. Previous experience with CP-nets and Design/CPN consisted of: 9 x 2 hour lectures introducing Petri nets, High-level nets and example applications, and a 2 hour practical session introducing Design/CPN.

The project member also had experience with SDL [CCI89], but lack of available software tools for analysis made it an unsuitable method for this problem.

A major component of this work was the exploration of different approaches to modelling IWS and the levels of abstraction that were appropriate. This process was slowed by the lack of experience with some Design/CPN features (eg. hierarchies). Including this initial work and reporting on results, the modelling and analysis consisted of approximately 11 man weeks of work.

4 Missile Simulation Model

4.1 General Structure

The model of IWS consists of five pages of CP-nets and a global declaration node. This is indicated in figure 4. A top level model of IWS shows the interfaces between the four simulation components (figure 6).

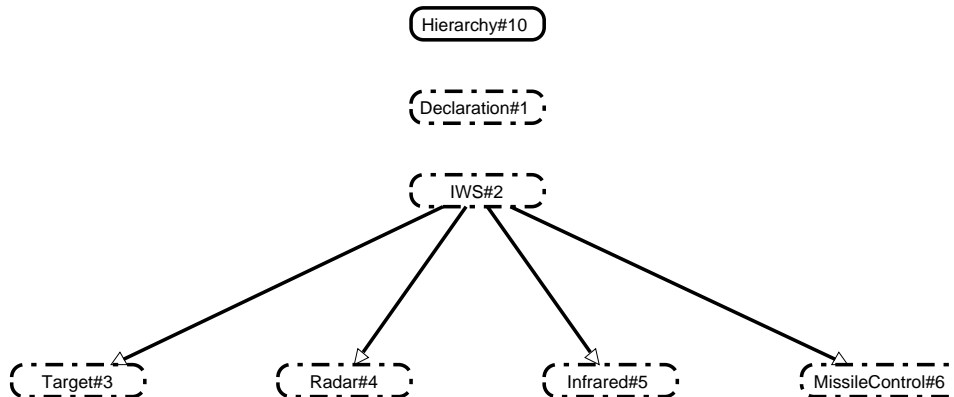


Figure 4: CP-net Hierarchy Page

Four transitions of the top level model are expanded into subpages to represent the simulation components ie. Target, Radar, Infrared, and Missile Control (figures 7 to 10).

4.2 Global Declarations

The global declaration page is shown in figure 5.

```
color Status = with Ready | Wait;
color Input = with Target_Delta | Pause | Stop;
color Motion = with Target_Trajectory | Missile_Trajectory;
color Range = with RF_Range | IR_Range;

var cmd:Input;
```

Figure 5: Global Declarations

The colour **Status** indicates the state of a thread when not processing data. For example, when the Radar thread (figure 8) is ready to receive new data from the target and missile, a **Ready** token is sent from place **Thread Ready**.

The colour **Input** represents a single input from the user. **Target_Delta** represents the change to target trajectory the user wants.

The positions and velocities of the target and missile are modelled as a trajectory. These are represented by the colour **Motion**.

The colour **Range** represents the range of the target relative to the missile. The radar (RF) and infrared (IR) ranges are different.

The variable **cmd** is used to represent the next user input. It can be either **Target_Delta**, **Pause** or **Stop**.

4.3 IWS

4.3.1 Top-level CP-net

The top level CP-net (figure 6) shows the functionality and flow of data in IWS. The user is not represented explicitly, instead the place `GUI inputs` stores the current command from the user (either a `Pause`, `Stop` or `Target_Delta`) and it is processed by one of the three output transitions. If the user issues a pause command the `Pause` transition fires and a new user command is generated. If stop is issued by the user then transition `Stop` will fire leaving no more user commands. If `Target_Delta` is the user command then it is sent to `Target` and the simulation continues.

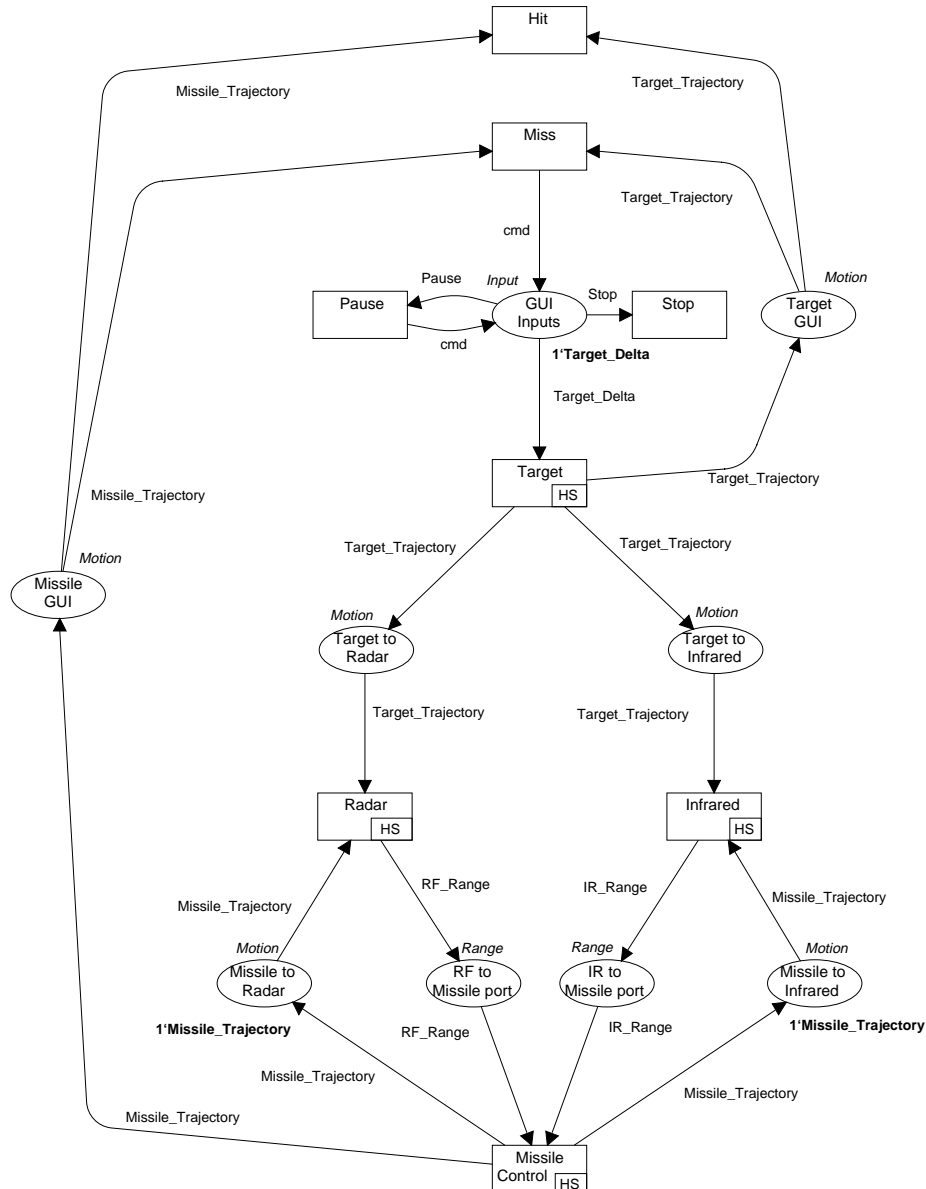


Figure 6: Top-level CP-net

When the new target and missile trajectories have been calculated two conditions can occur: the positions of each are the same (within a given tolerance) meaning the missile had hit the target and the simulation should stop; or the positions are not the same and

the simulation should continue. This is modelled using the two transitions `Hit` and `Miss`. If `Miss` occurs a new user command is generated and the simulation can continue. If `Hit` occurs, no more transitions will be enabled causing the simulation to stop.

4.3.2 Structure of CP-net Subpages

The four simulation component pages (figures 7 to 10) are structured into columns to visually separate the shared data, threads, processes, and communication mediums.

The first column represents the shared memory between each of the four threads. The data (or tokens) in this area can be accessed by any of the threads. When implemented, access to the memory will be guarded by the use of semaphores.

The second and fourth columns in the CP-net pages represent the procedures carried out by the threads and processes, respectively.

The third and fifth columns of the models represent the TCP/IP socket connections. The third column is for connections between the thread and its corresponding process, whereas the fifth column represents a connection between two processes. Although TCP/IP connections incorporate queueing mechanisms, they are modelled as simple buffers in the CP-nets. This is because the control of data flow does not allow overtaking of data, hence there are no need for queues.

For all the models, it is assumed that the socket connection has been setup and initial data for the processes has been received.

The functions performed by the simulation components in IWS are described in the following sections.

4.3.3 Target

The Target CP-net is shown in figure 7. The Target process begins in the “ready” state. This is indicated by an initial marking of `Target_Trajectory` for the place `Process Ready`. The first user command is also an initial marking for the place `GUI Inputs`.

When a user target change is received from the GUI, it is sent to the Target process. The transition `Calculate Target` represents the calculations made using the user target change and old target trajectory (from `Process Ready`) to produce a new target trajectory. This result is sent back to the Target thread and then to the GUI, Radar and Infrared.

4.3.4 Radar

The Radar thread begins in the “ready” state (figure 8). There is also an initial marking `Missile_Trajectory` in the `Missile to Radar` place. This marking represents the initial missile position and velocity so Radar (and Infrared) can calculate an initial range to be used by Missile Control.

When the target trajectory arrives at place `Target to Radar` (and the missile trajectory is in `Missile to Radar`), transition `Get Target data` is enabled. Upon firing, the target and missile trajectories are transferred together to the Radar process and with them a new range can be calculated. The range is sent back to Missile Control via the Radar-Missile socket, and also to the GUI via the Radar thread.

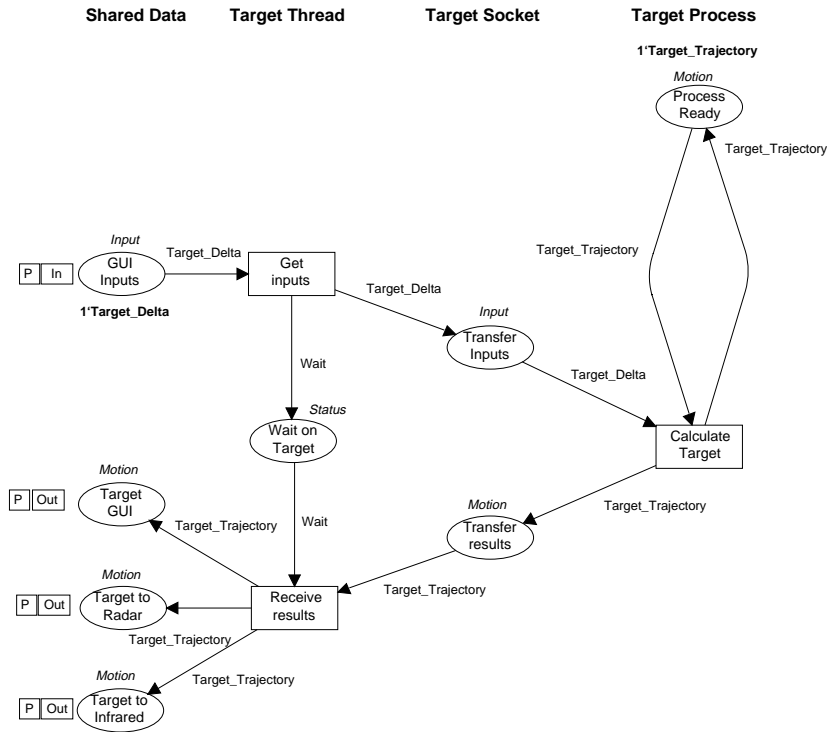


Figure 7: Target CP-net

4.3.5 Infrared

The CP-net model of the Infrared component (figure 9) operates in the same manner as the Radar model.

4.3.6 Missile Control

The Missile Control is straightforward (see figure 10). An initial missile trajectory is stored in `Process Ready` and when both the Radar and Infrared ranges arrive the new missile trajectory is calculated by the Missile Control process. The results are sent to the Missile Control thread which stores them as shared data to be accessed by the Radar and Infrared.

5 Analysis

The missile engagement simulation model was analysed using Design/CPN. Using Design/CPN interactive graphical simulations and automatic simulation modes are available. Simulating was useful when developing the models, and to view the flow of data in IWS. However, it was not suitable for testing all possible states of the models. The formal analysis of the CP-net model involved the OG tool of Design/CPN. The OG tool calculates the occurrence graph for the CP-net model. Rather than visually inspecting all nodes, queries were made of the occurrence graph to examine its properties. Table 1 shows the results of the occurrence graph generation.

The OG was cyclic - each simulation iteration was represented by one cycle of the occurrence graph. Figure 11(a) shows the initial marking (node 1) and its immediate successor and predecessor nodes. Note that the place and transition names used are

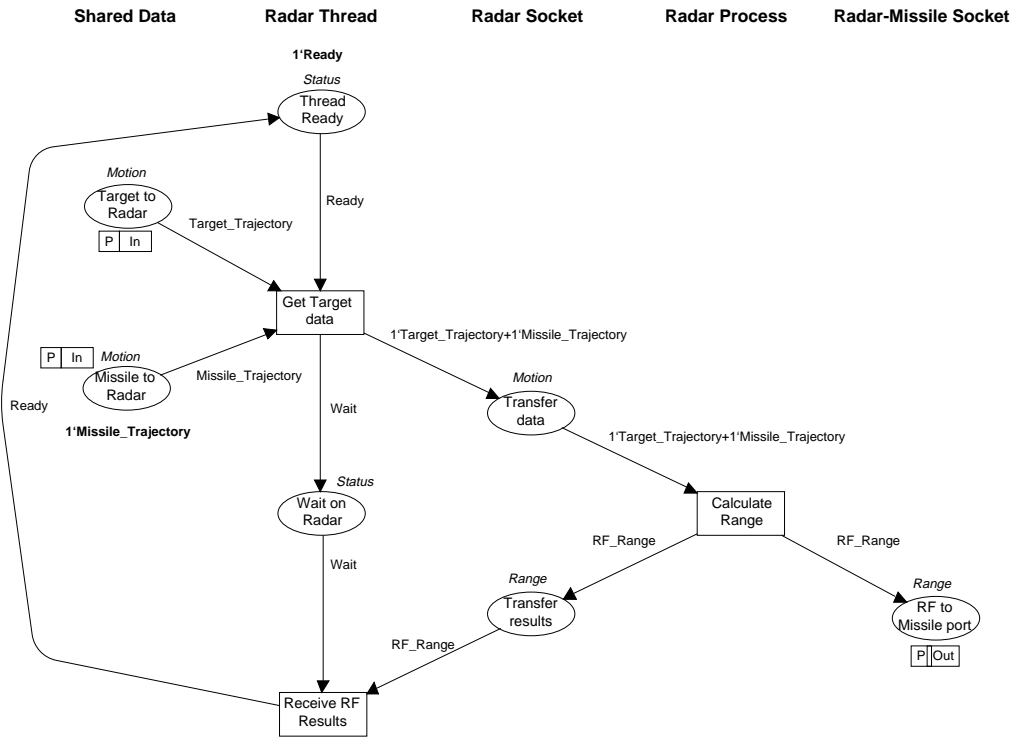


Figure 8: Radar CP-net

<i>Nodes</i>	<i>Arcs</i>	<i>Time</i>	<i>TerminalStates</i>	<i>Undesired</i>
57	124	1 s	1	0

Table 1: OG Results

extensions of the labels used in the models. Also, places with empty markings are not shown.

There was one terminal state and no deadlocks (undesired terminal states) for the model. Closer examination of the terminal state and the arcs leading to it confirmed that the simulation stopped by either a “hit” or by the user. Figure 11(b) shows the terminal state (node 43) of the OG. A selection of nodes leading to this node are also shown. The arc from node 31 indicates **Hit** has occurred causing the simulation to stop. The arc from node 41 indicates the user has input a **Stop** command again causing the simulation to stop. The other two input arcs of node 43 show the transitions that can occur after **Hit** or **Stop** occurs. These two transitions are **Receive RF Results** and **Receive IR Results**. When the Radar or Infrared threads receive results from their corresponding processes (ie. these two transitions fire), the data is made available to the GUI so the display can be updated. However, when **Hit** or **Stop** occur the display is frozen so the user can view and analyse the results. Therefore, if **Hit** or **Stop** occur before either **Receive RF Results** or **Receive IR Results** then the latest results will not be displayed. To be sure this does not occur, the GUI must force an update of the display when stopped before allowing the user to perform analysis on the results.

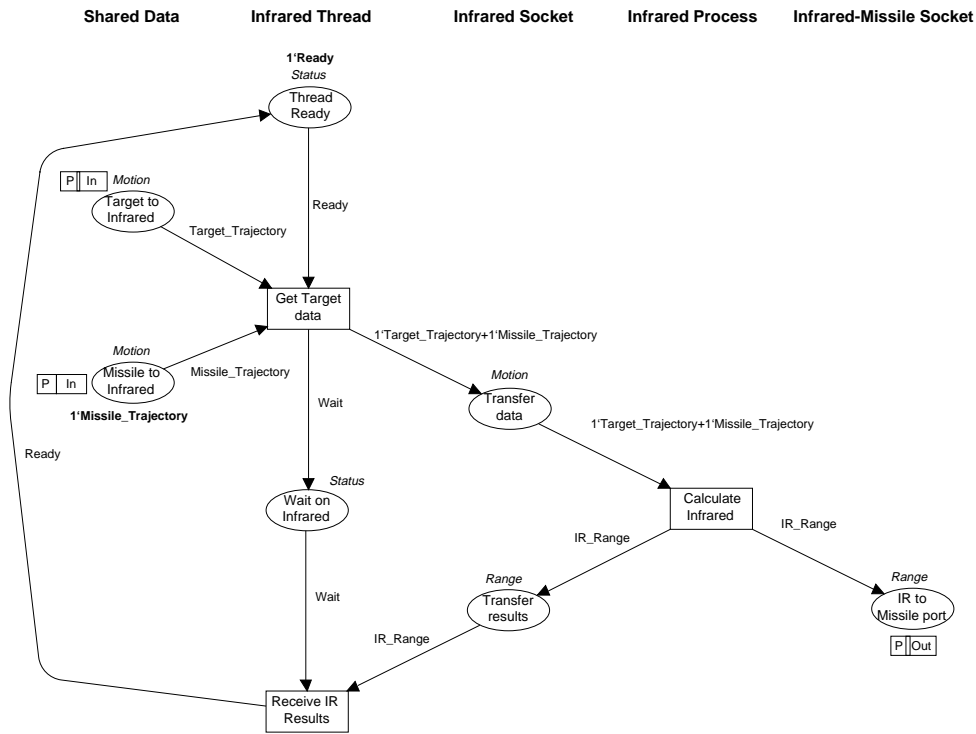


Figure 9: Infrared CP-net

6 Conclusions

Coloured Petri nets have been used to test the communication mechanisms for a distributed Air-to-Air missile engagement simulator.

The first step in achieving the goals was to create a top-level CP-net for IWS. This showed the data flow and basic functionality of the system. Using Design/CPN's hierarchy features, the relevant transitions were then expanded into subpages to model the interaction between threads and processes and the calculations performed by IWS. This allowed the concurrent execution of the IWS components to be analysed. This top-down approach to modelling IWS was advantageous because initially it was unclear how much detail was needed. Further functionality could be added when it is required.

Once the models were completed they were analysed using the OG tool of Design/CPN. This allowed occurrence graphs to be calculated and properties of it examined using queries. The analysis showed that the models behaved as required. There were no deadlocks and the individual threads and processes executed correctly. However, a new requirement of the GUI was identified, ie. a display update should be forced when the simulation is stopped.

From the analysis of the CP-nets, under the requirements from DSTO Australia, the communication mechanisms modelled are suitable for use in IWS. However, to take full advantage of the performance enhancement from concurrency and remote execution, it is necessary that the system performance is not restricted in other areas. In particular, a preliminary investigation into the effects of failures in the communication between processes and the methods for coping with such failures has been made. The models developed in this work served as a basis for this investigation.

There are also possibilities for improvements and further modelling and analysis to

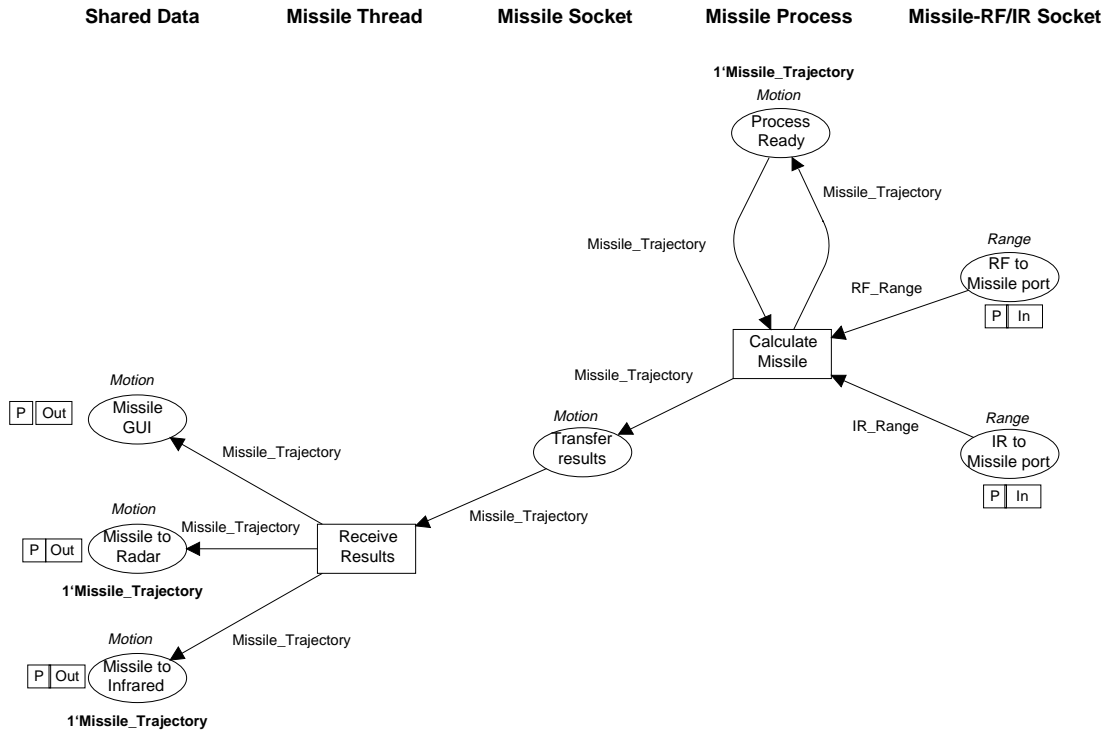


Figure 10: Missile Control CP-net

test other features of the IWS simulation.

- A requirements-level CP-net which specifies: interactions between the user of IWS and the simulation, and interactions between components within the simulation. A comparison can then be made between the requirements- and design-level CP-nets.
- Detail modelling of the GUI (at the design-level) and the interactions with the simulation. This would introduce more levels of concurrency (eg. the GUI updating it's display while the simulation performs the next calculation) and require more detailed analysis.
- Further modelling and analysis if multiple targets are introduced into IWS [CGW97a].

7 Acknowledgements

We would like to thank Hatem Hmam and Len Sciacca from DSTO for the technical background on Air-to-Air missiles and IWS, and Chris Steketee for his comments on TCP/IP. Also, the financial support of the University of South Australia's Institute for Telecommunications Research is appreciated.

References

- [CCI89] CCITT. Recommendation Z.100 Functional Specification and Description Language. In *Blue Book*, Geneva, Switzerland, 1989. ITU.

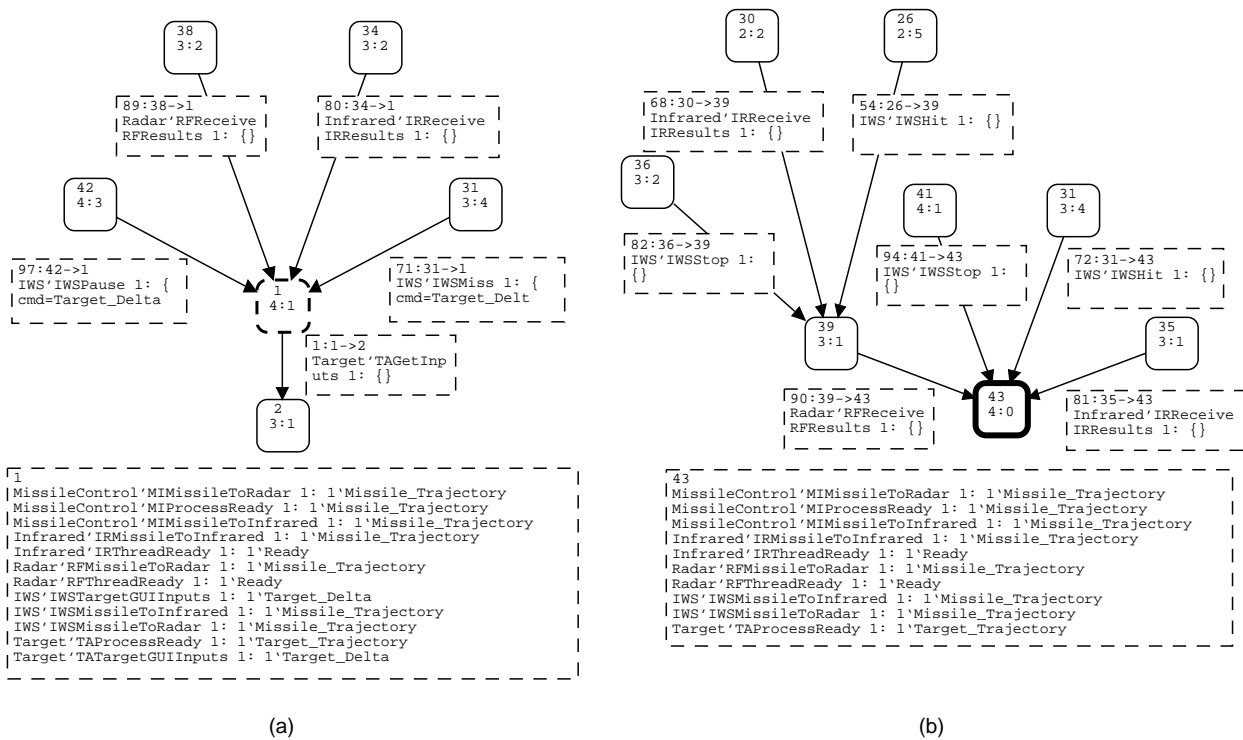


Figure 11: (a) Initial and (b) Terminal States

- [CGW97a] B. Collas, S. Gordon, and H. Widjaja. *IWS Requirements Specification*. Number 1.2. University of South Australia, Adelaide, Australia, 1997.
- [CGW97b] B. Collas, S. Gordon, and H. Widjaja. *IWS System Manual*. Number 1.0. University of South Australia, Adelaide, Australia, 1997.
- [CJ95] S. Christensen and K. Jensen. *Design/CPN Occurrence Graph Tool - User's Manual*. Computer Science Department, Aarhus University, Aarhus, Denmark, 1995.
- [Jen92] K. Jensen. *Coloured Petri Nets - Basic Concepts, Analysis Methods and Practical Use*. Springer-Verlag, 2nd edition, 1992.
- [Jen94] K. Jensen. *Coloured Petri Nets - Basic Concepts, Analysis Methods and Practical Use, Volume 2*. Springer-Verlag, 1994.
- [Jen97] K. Jensen. *Coloured Petri Nets - Basic Concepts, Analysis Methods and Practical Use, Volume 3*. Springer-Verlag, 1997.
- [JM96] J.B. Jorgensen and K.H. Mortensen. Modelling and Analysis of Distributed Program Execution in BETA using Coloured Petri Nets. In *Application and Theory of Petri Nets*, number 1091 in Lecture Notes in Computer Science, pages 249–268. Springer-Verlag, 1996.
- [Los95] P. Loshin. *TCP/IP for Everyone*. Academic Press, Chestnut Hill, MA, USA, 1995.
- [SG94] A. Silberschatz and P. Galvin. *Operating System Concepts*. Addison-Wesley Publishing Company, Reading, MA, USA, 1994.

- [Sof93] Meta Software. *Design/CPN Reference Manual for X-Windows*. Meta Software Corporation, Cambridge, MA, USA, 1993.
- [Tan92] A. Tanenbaum. *Modern Operating Systems*. Prentice-Hall, Englewood Cliffs, NJ, USA, 1992.