Proceedings of the 2001 Winter Simulation Conference
B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, eds.

THE RAPID MODELLING SYSTEM: A COMPONENT BASED APPROACH TO THE SIMULATION OF TACTICS

Phillip Martin

CORDA Ltd.
Apex Tower, 7 High Street
New Malden, Surrey, KT3 4LH, U.K.

ABSTRACT

A component based approach to the simulation and development of tactics or procedures was presented at WSC'99. This paper provides an update to the approach, describing the substantial progress made in developing a modelling tool set called the Rapid Modelling System (RMS) to take advantage of the original concept. The paper describes the problems encountered during the development and the methods employed to overcome them, whilst keeping to the overall aim of providing a generic structure to the RMS. The current functionality is described including the ability to use propagation data and target strength values for sensor performance. An illustrative example of a tactical set of procedures is described and a worked example is provided showing how the RMS allows variations to be made in a controlled and repeatable manner. The RMS is written in EXTEND ™ (Imagine That Inc).

1 AIM OF THE DEVELOPMENT

The aim of this development was to create a desktop tool that could be used by analysts to investigate tactics in an intuitive way.

2 INTRODUCTION

The concept of a method for the investigation of tactics using a component-based approach was presented by Martin (1999).

In this paper the Rapid Modelling System (RMS) is described, which has been developed using this concept. Although a number of problems occurred during the development, the overall concept has proven to be robust providing a useful and flexible tool. The RMS has been used in a tactical development study as well as a more general simulation where tactics are involved.

2.1 What is a Tactic ?

The term tactic may mean different things to different people. In this paper a tactic is a series of instructions and or processes which are stepped through, driven by external stimuli. The path taken through these steps may vary according to the particular stimuli present or the order of the stimuli, but all the options are defined prior to the simulation. The tactic is the set of instructions or the orders to be followed.

It is the basis of the concept that complex tactics can be built up from a generic set of simple building blocks.

2.2 The RMS Development

A common problem of simulations is that they are designed to model the performance of systems and where tactics are an input. The tactics are embedded in the model code and to change tactics beyond the level of parameter changes involves altering the code with the inherent overhead of testing and validation.

The RMS was to allow development and changing of tactics to be undertaken by the analyst, who is computer literate but does not generally have the time or inclination to learn another detailed software product.

Software products such as EXTEND have addressed similar problems for general simulation by providing operating systems in which models can be developed using libraries of blocks of code combined together using drag and click techniques. The RMS applies the fundamentals of EXTEND to the particular problem of tactical development by allowing the analyst to build up tactics from generic building blocks and providing an underlying model in which to simulate the effects of the tactics.

The RMS consists of two distinct parts, which are now described:

- The underlying model,
- The tactical representation.

3 THE UNDERLYING MODEL

The underlying model consists of units representing platforms and sensors and a representation of the world in which they exist.

3.1 Representation of the World

The model is based around a structure of environment through which the units communicate and react.

- The physical environment. This is the "real world" space into which the units are placed and within which they move. This environment keeps track of the units' positions. It maintains the absolute values, whereas the units themselves may be provided with data to which errors have been added. The physical environment has a flat earth (x-y-z) structure.
- Sensor environments. The units interact with the "Real World" through these environments Within a particular environment, a unit can have:

    1) Active sensors, which radiate into the environment and process return reflections,
    2) Reflectors, that reflect the radiation,
    3) Modifiers that return a modified signal,
    4) Passive receivers that receive signals,
    5) Generators that generate radiation/ noise continuously.

It is only through the use of their sensors in the different environments that the units are aware of each other.

Examples of sensor environments, which all have similar structures and operations are:

- Sonar,
- Radar,
- Visual,
- IR.

A specialist sensor environment is the Communication Environment. This differs from the other sensor environments because of the variety of messages that could be passed. These do not mirror the verbal communication, but provide the information and triggers that would result. However, the counter detection of these messages is possible as is the ability to switch on / off receivers.

Units are capable of logging into and out of the environments as the simulation proceeds. Units that are registered with the Physical Environment are "in" the simulation. This registering / de-registering can also happen within a simulation to represent the creation or destruction of units.

3.1.1 Signal Propagation

Initially detection probability was given a "cookie cutter" representation. More recently this has been modified to allow the propagation of the signal through the environment to be calculated and the relative aspect of the sensor and target to be taken into account. This has been achieved by providing look up tables generated externally by a detailed propagation model and the appropriate values being extracted by the environment as the simulation takes place. However this facility would not be appropriate for all simulations so it can be switched off when necessary. This main aim must be to keep the model in balance, in terms of the fidelity of the input data and the calculations being undertaken using that data.

3.2 Units

The units are made up of blocks, contained in libraries, which allow the units to interact with the environments, Figure 1.

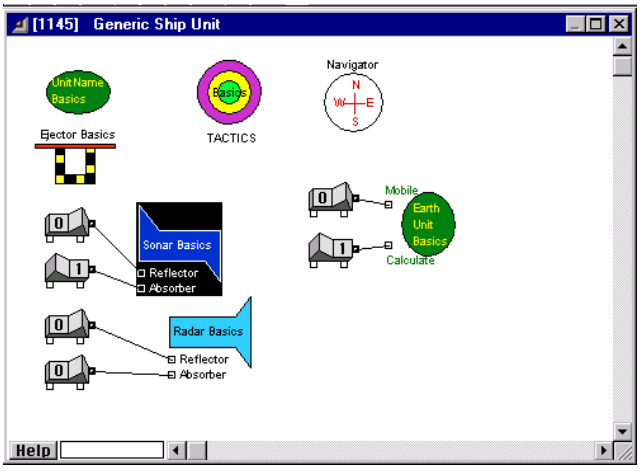

Figure 1: Structure of a unit

This unit has two sensors, a sonar and a radar, a tactical block and the blocks required to interact with the underlying model. The majority of these blocks are hierarchies and contain further structures themselves. For example the sonar, Figure 2.

Within the sensor hierarchy, the transmissions are passed as messages (items) to the appropriate sensor environment where they are processed and returned as necessary. A variety of information is returned with the message which is then used by the receiving sensor to process the data

Within the navigator block, the unit manages parameters such as course, speed rate of climb, which define how it moves about the physical environment. The unit does not control where it is in the world. That is done by the physical environment using the parameters passed to it from the navigator block.

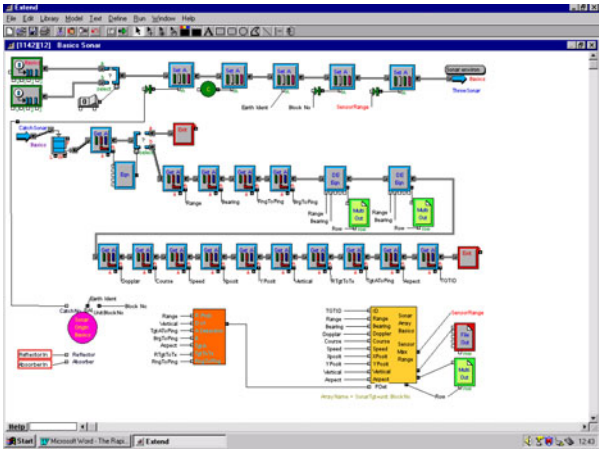

Figure 2: Structure of the Sonar

3.3 Scenario Set-up

The analyst interacts with the model through EXTEND's working window. Figure 3. This is scaled by the model to have real world dimensions. The units are placed in this window to set up the scenario for the simulation and the positions within the window are used as start positions. Additional blocks can be placed in the window to represent such things as search areas or route waypoints with which the units can interact.

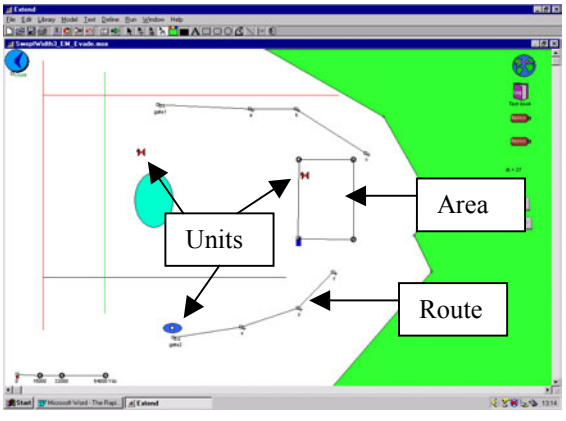

Figure 3: Model Window

The unit icons can be set to move about this window to mirror the movement of the units within the physical scenario. This facility is very useful for presenting the effects of tactical changes although it does have a cost of increased run time. This movement can be switched off for batch running.

4 TACTICAL REPRESENTATION

The representation of tactics is built up within a hierarchy contained within the unit structure. This Tactical hierarchy contains almost exclusively custom blocks, which can be grouped into 4 types, Figure 4.

- *Tactical blocks*, that make up the tactics,
- D*ecision blocks*, in which the steps or logic of the tactics are generated,
- *Controller Blocks,* that control when each part of the tactics is to be used,
- *Helper Blocks,* that extract / store information to allow the decider blocks to make the decisions.

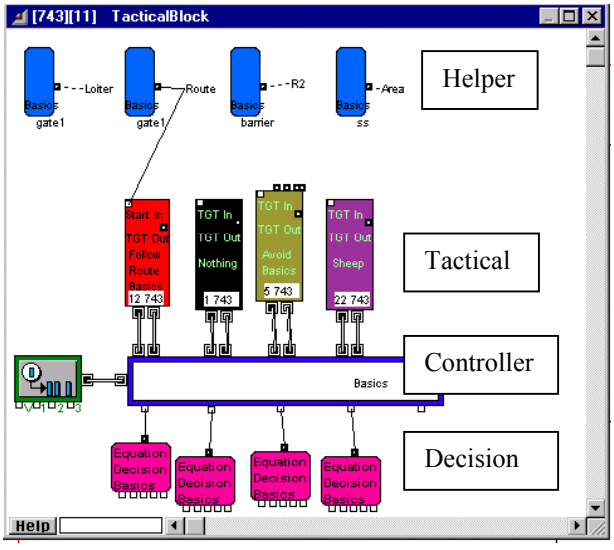

Figure 4: Tactical Hierarchy

It is in this tactical structure that the principles of EXTEND have been modified for the RMS. In EXTEND the simulation progresses by messages or items being passed between blocks. But whereas this occurs in the standard EXTEND in a flow like process from start to finish, in the Tactical hierarchy only one item is generated and this is passed between the Tactical blocks like a relay baton.

In this Tactical hierarchy any number of Tactical blocks can be combined. Unlike other methods which employ a flow diagram like structure to link the blocks and where all the links have to be pre-set, in the tactics all the block are effectively linked to all other blocks via the Controller blocks. The actual links used are created as and when they are needed.

Which particular block is operating at a specific time is controlled by the decider block. This has a dialog in which the operator can specify when the tactic is performed (the logic), Figure 5.

In this example the trigger occurs when the target range is less that 3000. When the logic is true, i.e. the Result = 1, a message is passed to the controller block to move the "relay baton" item to the selected Tactical block.

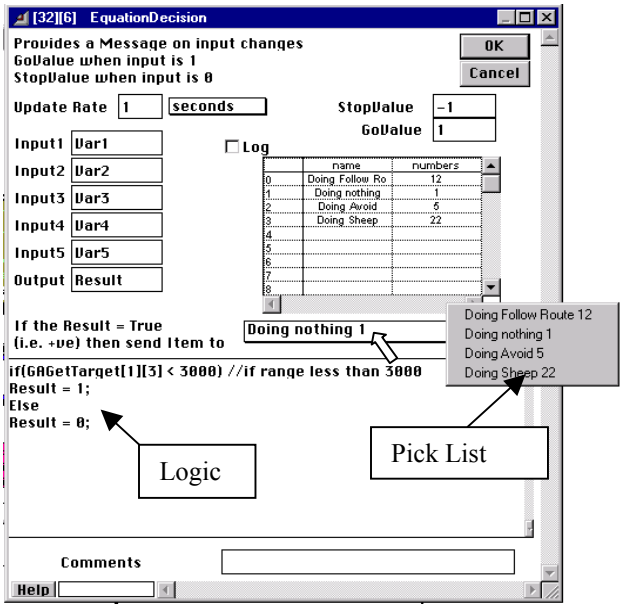

Figure 5: The Decider Block Dialog

The choice of Tactical block is selected using the pick list, also shown in Figure 5, which appears automatically when the mouse pointer is placed upon the dialog parameter. This list contains all the Tactical blocks in this unit and is automatically updated if additional blocks are dragged across from the Tactical library. Therefore the particular tactic can be changed simply using the computer mouse.

Any number of decision blocks can be used which may use the same Tactical block for different conditions.

Within the simulation, if the conditions set up in the decision blocks are met then a message is sent to the Controller to start the chosen tactic. This is achieved by passing the Item (relay baton) from its current position to the chosen tactic.

The decision block does not need to be told where the item is; it merely signals the next step. With this arrangement the tactics are reactive to the conditions within the model.

This could be:

- time based, where a trigger occurs at a set point in the simulation,
- an effect of sensor contacts such that a tactic occurs when a target is found,
- a unit parameter, a tactic is triggered by a course change or fuel limit being reached,
- because a previous tactic has now finished.

The conditions and the tactics have to be set up by the analyst, but additional tactics can be brought in or the conditions changed without affecting the existing set-ups.

The effect is to have total flexibility in how the tactics are set up, but without resorting to extensive coding changes.

One of the particular helper blocks is the State block. With this block the user can specify the various states in which a unit could find itself during a simulation. This block is a focus for state changes and it maintains a record of changes for later analysis. Tactical blocks can be set up to react differently depending upon the state and the current state is extracted from the State block when required.

5 WORKED EXAMPLE

In this section a simple example is presented to show how tactics can be built up from distinct stages and how this can be easily represented within the RMS.

5.1 Model Set-up

The example is a barrier search by an ASW ship and the attempted penetration of that barrier by a submarine. The ship is following a “bow- tie” barrier pattern and the submarine moves towards the barrier from its starting position shown in Figure 6.

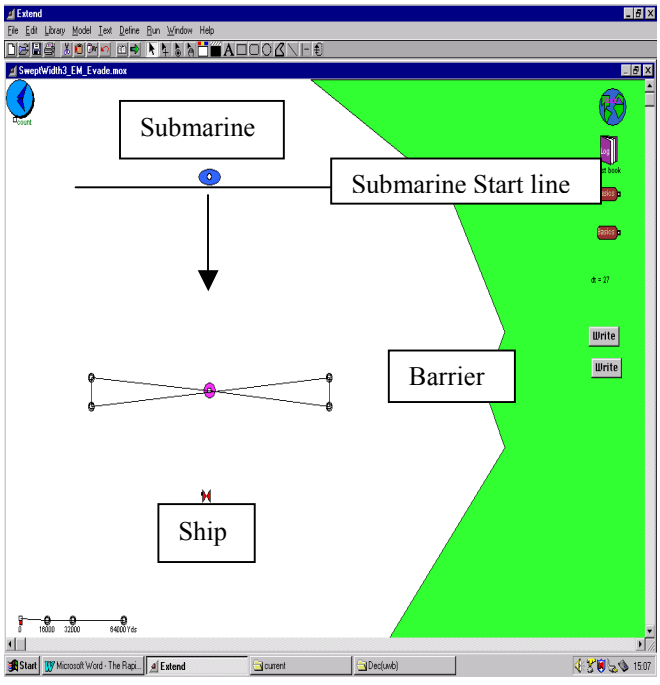

Figure 6: Model Set-up

The ship unit is set up to have an active sonar by placing the general sonar hierarchy within a generic ship unit and setting the parameters for an active sonar. The submarine is given the same general sonar hierarchy but with the parameters set to be a sonar reflector and also a passive receiver. These RMS hierarchy structures are show in Figures 7 and 8 for the ship and submarine.

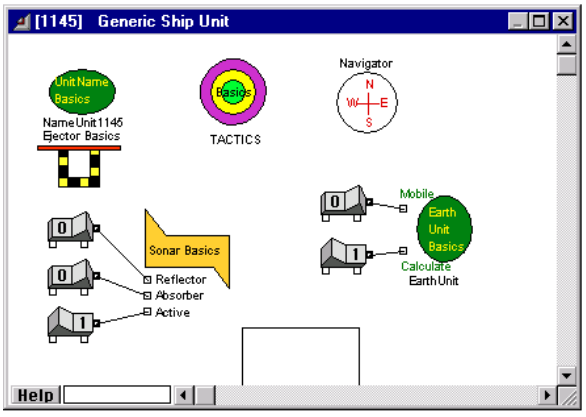

Figure 7: Ship Unit Hierarchy

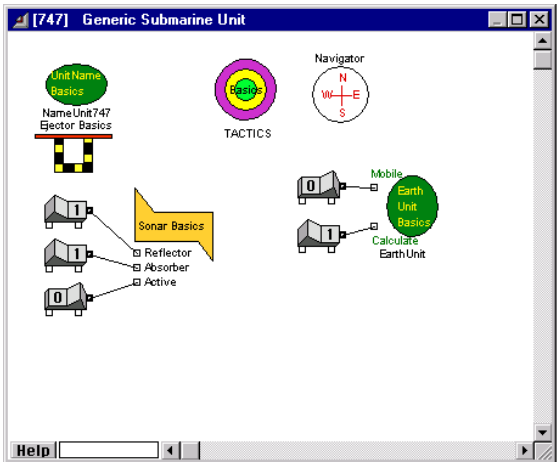

Figure 8: Submarine Unit Hierarchy

5.2 Initial tactics

To start, the submarine is set to have a basic tactic of crossing the barrier in a straight line. The state transition diagram for this is shown in Figure 9.

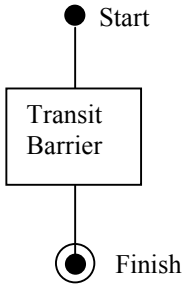

Figure 9: Initial Tactics

This basic tactic is created in RMS using one block of code called "Transit", which is placed in the Tactical hierarchy as shown in Figure 10. This block is then selected in the decision block dialog, shown in Figure 11. The decision logic is extremely simple being "just to do this tactic".

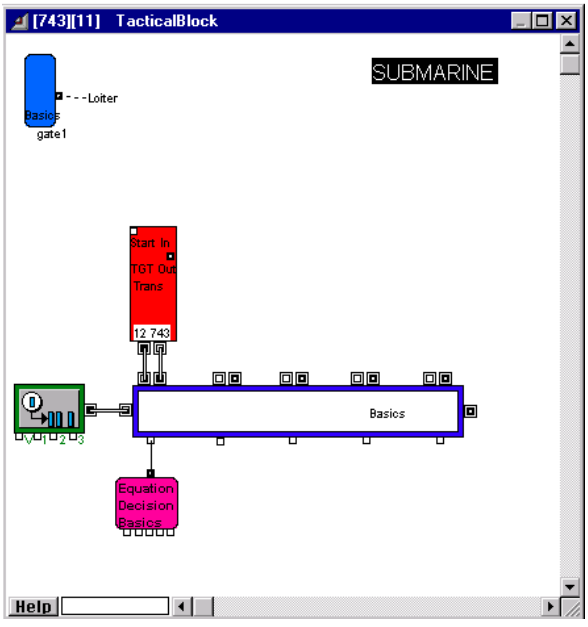

Figure 10: Initial Tactical Set-up.

5.3 Developing the tactics

As the ship patrols the barrier, it is actively searching for the submarine. The submarine receives these signals passively and can be made to react. The submarine is given the tactic that when the range R to the ship is less than Y then avoid the ship by going around it with a minimum separation distance X, as shown in the state transition diagram Figure 12. Note that while performing the avoidance tactic, if the range to the ship opens to beyond Y then the submarine will return to the transit behaviour.

It is possible for the submarine to find itself within the distance X so another part of the tactic is created to "Play Dead", that is, to point towards the ship and to slow right down. When the distance increases again then the submarine continues with the previous activity. Because the ship is not transmitting continuously, the "Play Dead" could be required from both the transit or the avoid behaviour. Figure 13.

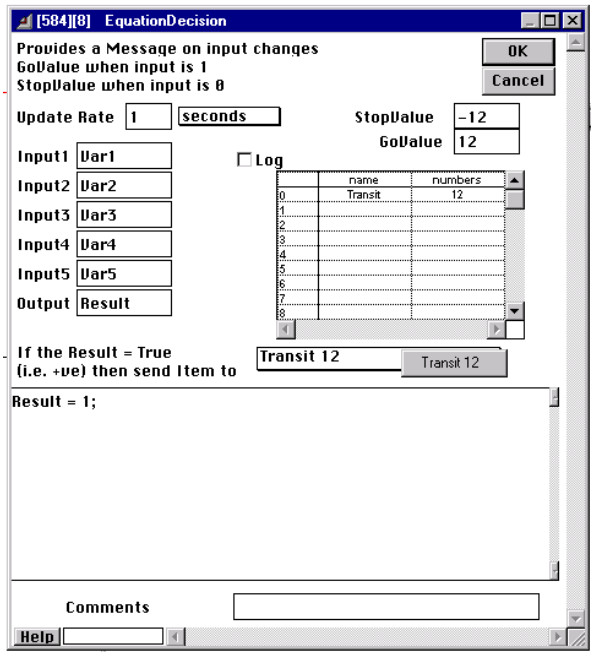

Figure 11: Decision Dialog

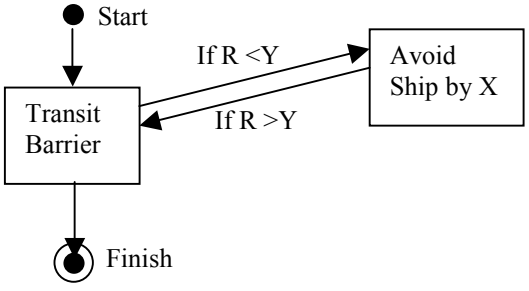

Figure 12: Submarine to Avoid the Ship

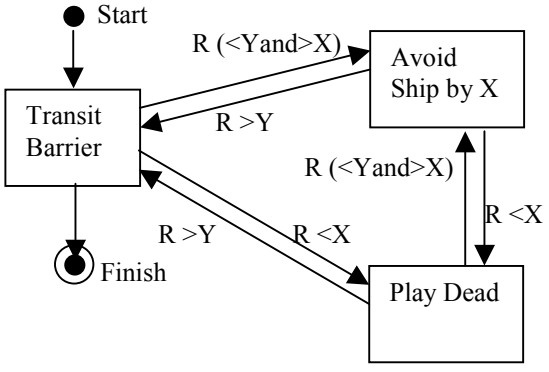

Figure 13: Addition of "Play Dead"

Within the RMS Tactical hierarchy, two additional blocks, for the Avoid and the Play Dead, are added from the library, together with corresponding decision blocks to select them when the conditions are met. In Figure 14 the Tactical hierarchy is shown with a dialog for one of the additional decision blocks in Figure 15.

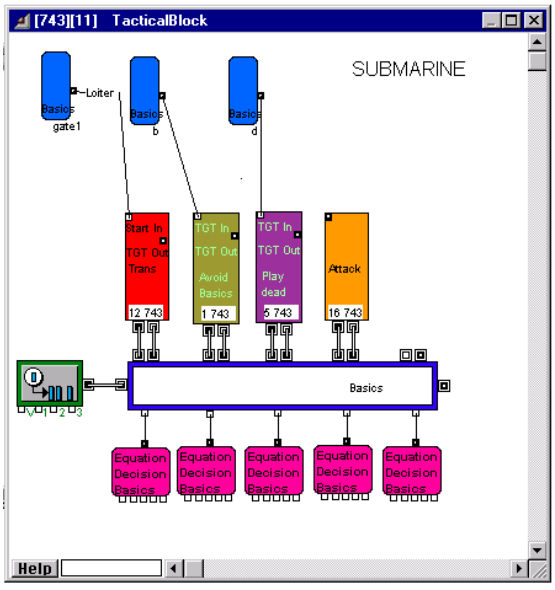

Figure 14: Tactical Hierarchy.

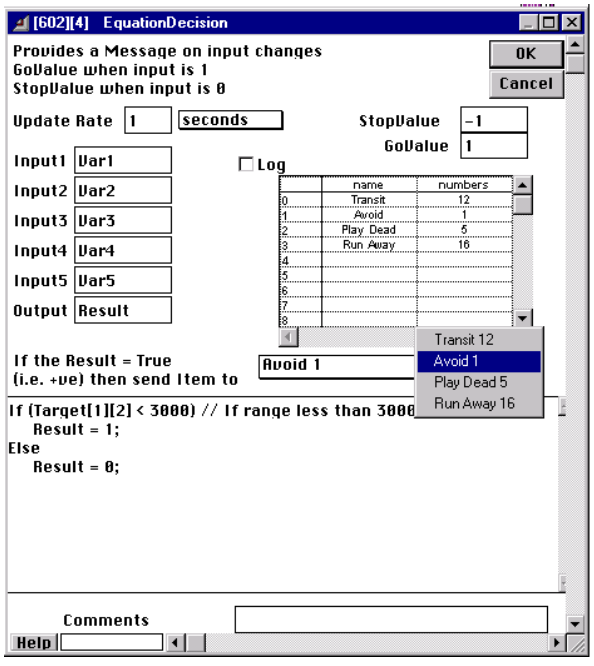

Figure 15: Multiple Dialog Choice

5.4 Ship Attack

Within the RMS all units can have independent tactics, so it is possible, with this example, to have the ship react to its own sonar contacts and attack the submarine. The sub-

marine can also react to this attack by say running away. Figure 16.

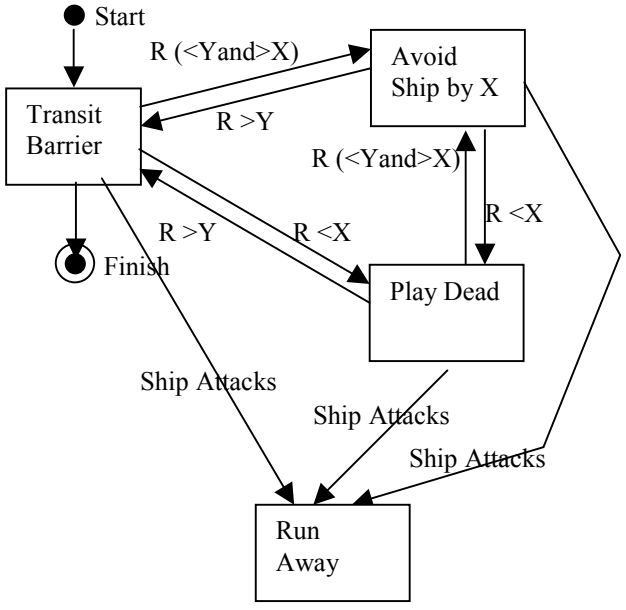

Figure 16: Ship Attacks Submarine

This again is simply another Tactical block with associated decision logic placed into the Tactical hierarchy. With all the Tactical blocks, what the unit does in detail is programmed into the block itself and modified by parameters within the block dialog. The modularity of the design is such that if a change of tactic was required this can be done by simply changing the selection of the blocks in the decision dialog.

For example, the submarine tactics are to change when the ship range R is less than X. The submarine is to attack the ship instead of playing dead, Figure 17.

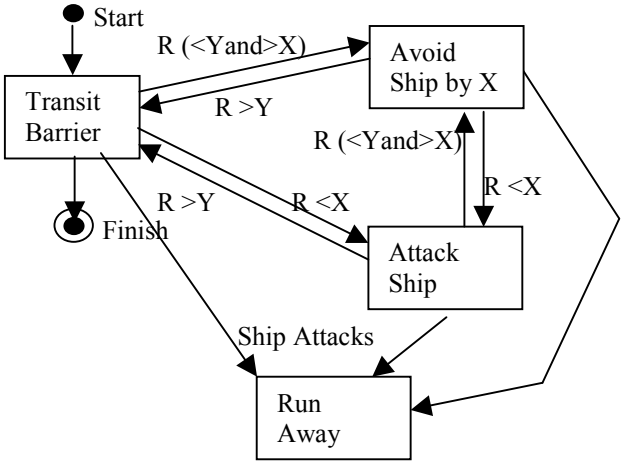

Figure 17: Submarine Attack the Ship

In the Decision dialog, which was selecting the play dead tactic, the user can now select Attack from the pick list which is automatically updated after Attack has been brought down from the Tactical library, Figure 18.

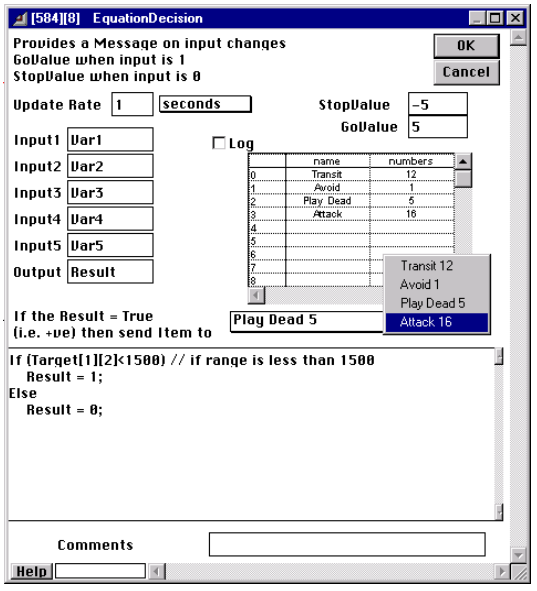

Figure 18: Selecting Submarine Attack

5.5 Running the Model

The model is run, and the first step of the submarine tactics is initiated, that is the transit. The ship produces sonar transmissions that are passed into the sonar environment, where return signals are generated off the submarine (as a sonar reflector) and returned to the ship as well as the active transmission being given directly to the submarine. The Sonar Hierarchies calculate if the signals are strong enough for a detection to occur or not.

The submarine reacts to its detections by “avoiding” or “playing dead” depending upon the range to the ship. This occurs as the decision blocks initiate the appropriate tactic, defined by their decision logic. The simulation continues with the Tactical blocks being switch on or off as the relative movements and positions of the units change.

5.6 Expansion of the Model

Beyond this simple example, the model can be set up with any number of units that can interact with each other. For example, the ship could operate with a helicopter, which it can launch to investigate the sonar contact. The helicopter is a unit, which when it is launched it registers with the physical environment, and it becomes part of the simulation. The helicopter itself can have sonobuoys which its drops. Again they are units and have passive or active sonars, which operate in the sonar environment. Contacts can be passed back to the helicopter as mes-

sages which the helicopter reacts to using additional sets of Tactical blocks. For example by dropping a torpedo.

The submarine can be set up to react to all these events.

These tactics and reactions are all set up using the library of blocks, defining the reactions in the decision blocks and setting up parameters in the block dialogs.

Where new Tactical blocks are needed formal coding has to be undertaken to create them, and the block design has to conform to the requirements of the RMS, but this can be undertaken by programmers to meet the requirements of the analyst. Once created the blocks can be stored in the Tactical library and used by the analyst as required.

6 PROBLEMS ENCOUNTERED DURING DEVELOPMENT

The aim of the development was to generate generic blocks that could be built into any tactical definition. The main problem was to make them general enough without

- requiring so many parameters that they became unwieldy ,
- so simple that large numbers of blocks are needed to perform simple tasks.

A related problem is the definition of when the tactic starts and finishes (for example if the tactic is to search an area is the going to the area part of the tactic?).

Although this is really an on-going problem, the solution found to be most convenient is to have generic blocks that do specific tasks, but then set up customised versions in libraries for different conditions. This then allowed the customised blocks to be imported into models without the need to set up the full set of parameters each time.

The unit undertaking a particular mission will usually go through different states or conditions as the mission progresses. The basic tactical steps may well be different depending upon the state the unit is in. To avoid having duplicated Tactical blocks with slightly different parameters, a specialist helper block was developed to record and keep track of the state of the unit. By having a defined specialist block, it made the task of referencing and communicating from other blocks automatic.

The interaction between the Tactical blocks and the unit functions such as speed height etc. were initially set up using the EXTEND connections between blocks. But as the tactics became more complicated it very quickly became necessary to have a more automated solution. This was achieved by setting up unit global arrays, which could be accessed by any block within the unit. Additionally this change allowed the building of a unit to be more automated as connections do not have to be made. The down side is the lack of visibility as to what information blocks are using.

As the tactical representations have become more complicated, it is apparent that having all Tactical blocks as stand alone sets is not appropriate. It needs to be possible to generate a more complicated block from a number of sequential simple blocks, rather than creating one super-block. A solution is to create a shell block hierarchy which is seen by the Controller as a single block, but contains inside itself a combination of blocks in its own right. This internal combination can be set up by the operator in the same way as the standard blocks.

The general design of the RMS provides an easy and convenient way of studying tactics, but the very open nature of the simulation environment provides one of the biggest problems. That is the problem of configuration control and build standards. The problem can be alleviated by locking libraries and not letting the analyst have access to the source code. But this goes against the open philosophy of EXTEND and the RMS development aim. The solution is to follow the EXTEND example where the "operating system" is hidden (in the RMS case the underlying model) and provide appropriate checks as blocks are created and at the start of runs.

7 CONCLUSIONS

The general concept described in Reference 1 has proved to be sound and a working system has been developed.

There have been a number of problems relating to the tactics becoming more complicated, but modifications to the implementation have overcome them. Configuration control remains the main problem due to the very flexible nature of the modelling environment.

The RMS is being used to model tactics and as a general simulation of military units.

REFERENCES

Martin, P. L. R. 1999 The Modelling of Tactics and Procedures Using a Component Based System. *In Proceedings of the 1999 Winter Simulation Conference* ed P. A. Farrington, H. Black-Nembhard, D. T. Sturrock, G. W. Evans.

AUTHOR BIOGRAPHY

PHILLIP MARTIN is a senior analyst and consultant within the Centre for Operational Research and Defence Analysis (CORDA), a wholly owned subsidiary of BAE SYSTEMS. He is a graduate of Aeronautical Engineering from Bristol University, and has been an Operational Analyst for over twenty years. During this time he has been involved with projects for all the UK Armed Services. He is currently responsible for the front line analysis undertaken by CORDA staff for the UK MoD and was himself a front line analyst working for the Maritime Warfare Centre (Fleet OA Staff) for six years. His e-mail address is <phillip.martin@corda.co.uk>