AR-010-226

DSTO-TR-0533

JOSE – Joint Operations Simulation Environment

M. Iob and D.A. Craven

DEPARTMENT ◆ OF DEFENCE

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION

JOSE - Joint Operations Simulation Environment

M. Iob and D.A. Craven

Air Operations Division
Aeronautical and Maritime Research Laboratory

DSTO-TR-0533

ABSTRACT

JOSE has been developed for use in operational effectiveness studies performed in support of the acquisition process for Project AIR 5077 Airborne Early Warning and Control (AEW&C). It can be used to simulate an AEW&C and the Australian environment it must operate in. This includes platforms, sensors, C3 structure and other ADF assets that may be involved in joint operations. Development of JOSE will continue to meet the needs of the acquisition process and to support the capability when it enters service.

DEPARTMENT OF DEFENCE

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION

Published by

DSTO Aeronautical and Maritime Research Laboratory
PO Box 4331
Melbourne Victoria 3001

Telephone: (03) 9626 7000
Fax: (03) 9626 7999
© Commonwealth of Australia 1997
AR-010-226
May 1997

JOSE - Joint Operations Simulation Environment

Executive Summary

Project AIR 5077 aims to acquire an Airborne Early Warning and Control (AEW&C) capability for the ADF. DSTO has had significant involvement in this project from its inception, providing support for the acquisition process. In AOD much of the work has been in the form of operational effectiveness studies to determine the level of capability required by the ADF.

It was clear from an early stage that a model would be required to ensure that the studies could be completed within the project time scales. JOSE was developed to meet this requirement. It can be used to simulate an AEW&C and the Australian environment including the platforms, sensors, C3 structure and other ADF assets that may be involved in joint operations with an AEW&C.

To date JOSE has been used to study the various AEW&C options on offer in representative scenarios, and measures of their operational effectiveness were extracted. These results have contributed to the process of determining the levels of capability required in the AEW&C. JOSE will continue to be developed to meet the needs of the acquisition process and to support the capability when it enters service.

Authors

M. Iob
Air Operations Division

Mauro Iob graduated from Royal Melbourne Institute of Technology with a degree in Aeronautical Engineering in 1985. He also has a Graduate Diploma in Digital Computer Engineering from Royal Melbourne Institute of Technology. In 1985 he joined the Aeronautical and Maritime Research Laboratory and has worked in the field of simulation of aircraft, aircraft systems and aircraft operations. This has involved development of F/A-18 flight control system model, a cockpit research simulator and more recently the JOSE model for studying AEW&C operations.

D.A. Craven
Air Operations Division

David Craven obtained his BEng in Digital Systems and Computer Engineering from the Royal Melbourne Institute of Technology. He joined Systems Division (now Air Operations Division) in 1988, and has worked extensively in the fields of real-time computer simulation and human interface design.

Contents

1. Introduction

Project AIR 5077 aims to acquire an Airborne Early Warning and Control (AEW&C) capability for the Australian Defence Force (ADF). DSTO has been involved in this project from its inception providing support for the acquisition process. A large proportion of the work done in AOD has been in the form of operational effectiveness studies. It was clear at an early stage that a model would be required to enable operational effectiveness studies to be performed as required by the project time scales.

The model that has been developed to meet this need is the Joint Operations Simulation Environment (JOSE). The model can be used to simulate an AEW&C and the environment that it will operate in under Australian conditions. This includes the platform, sensors, C3 structure and other ADF assets that may be involved in joint operations.

JOSE was used for the Project AIR 5077 Project Definition Study: Phase 1, Part 2. In this work the various AEW&C candidate options were modelled and measures of their operational effectiveness were extracted from the simulation results. The details and results of this analysis are described in References 1 and 2.

This report describes the current state of development of JOSE and the approach used in its development with some discussion of its future. Section 2.0 and associated subsections describe JOSE functionality and its design as well as the plotting program developed to process JOSE output. Section 3.0 details the software implementation issues associated with the development of JOSE. Then Section 4.0 and its subsections contain details of how JOSE is used and the results obtained from it. To illustrate the way JOSE operates a generic simulation output is described. The areas for the future development of JOSE are discussed in section 5.0. Finally section 6.0 contains some concluding remarks regarding current and future development of JOSE.

2. JOSE Functionality and Design

JOSE was developed as a tool for performing operational analysis studies in support Project AIR 5077. The functionality incorporated in the model was driven by the needs of this work, however software design and development was kept as general as possible. Consequently it is considered that the model has applications in other areas of operational analysis work for the ADF. The functionality and approach to design are described in the sections 3.1 to 3.3 below. An issue in understanding the results obtained from JOSE was visualization of simulation runs and plotting of the output. The graphical display and plotting program developed to support these tasks are also described in some detail in sections 3.4 and 3.5 respectively.

2.1 General Description

JOSE can be used to model the Australian Air Defence System (AADS) and other Defence assets which may be involved in joint operations with it. The scope of the model is limited to a representation of what is termed here as an Area Of Operations (AO) with a blue and red side. Those elements of the ADF that participate in the modelling are termed 'entities'. An AO is assumed to contain the following entities:

- A blue AO commander who has ultimate control of all the blue entities;
- Blue entities consisting of:
    - At least one AEW&C;
    - Fighter interceptors on ground alert;
    - Ground based radars;
    - Other blue entities with or without sensors;
- The C3 structure linking all blue entities;
- Red threat entities.

All the entities navigate with reference to a curved earth coordinate system and all electromagnetic emissions are affected by line of sight (LOS) and range limitations as appropriate. Thus for instance a microwave radar will be affected by LOS and range limitations whereas a High Frequency (HF) radio will only be affected by any range limitations.

All blue entities communicate either by a data link or via voice communications. The data link is modelled as a network of participating units where each is polled in turn to transmit data while all other units listen. Consequently there is a network cycle time associated with this process. This is very similar to the way the Link11 Tactical Data Link operates in reality (Reference 3). Other data links such as Link16 (Reference 4) which operate essentially instantaneously can also be modelled by setting the cycle time to zero. Voice communications are modelled by assuming that a message takes a certain amount of time to be transmitted and reception is delayed by that amount of time.

The behaviour of an entity reflects the behaviour of a corresponding ADF element; eg. an AEW will exhibit different behaviour to a fighter aircraft. This behaviour is coded into JOSE and can be controlled through the input data for the model. Broadly, entity behaviour can be described as follows. The initial motion of all entities, if they are moving, is to follow a route defined by a set of waypoints generated from data in the input file. They will only vary from this if they are commanded to or on the basis of some information received from their sensors or from communication links. The process of coming to some decision to take an action is modelled by a fixed delay.

The behaviour of red entities is currently very simplistic. They launch strikes from their bases which have a predefined flight profile and target and they do not deviate from this during the scenario. They do not react in any way when they are attacked by blue forces. Red forces have no Air Defence Ground Environment (ADGE) or any wide area surveillance capability. This has been sufficient for the studies performed to date, however future studies may require that the red forces possess these capabilities. This involves instantiating a similar structure for the red entities as currently possessed by the blue forces. However, red forces would not necessarily exhibit the same behaviour as the blue forces.

Currently scenarios are input into JOSE via script files which can be read and edited by the user. This process is described in detail in section 4.1. In these scenario files the user defines all the entities (ie. AEW, fighters, commanders etc.), their sensors and the relationships between the entities. JOSE was developed to perform Monte Carlo simulations for operational analysis studies. In these simulations one or more initial conditions are varied randomly within some range over many individual runs to determine the sensitivity of the result to these initial conditions. Any one of the runs can be replayed if required. This technique was used extensively for AIR 5077 related studies however the model can also be used to perform single runs of a scenario.

2.2 Model Design Philosophy

The use of Object Oriented programming in developing this model is very appropriate because it is not difficult to view the AADS as a collection of objects. Whilst an adhoc Object Oriented design technique has been used in developing the model there is an overriding principle driving the design. This simply stated is that the simulation should as much as possible "mimic" or model the real world in all respects.

There is any number of ways in which to abstract the AADS into objects. The approach used here was driven by the principle mentioned previously. As a result there is the concept of an Environment object which controls and affects how entities in a simulation interact. All entities in the simulation are composed of a platform object and other objects such as sensors, communications links etc. which can be attached to the platform. For instance an AEW entity will consist of a platform object, radar object, communications device object and conceivably other objects as well. These concepts have been used to develop the detail design of objects used in the simulation, as described in the next section.

2.3 Model Design

The structure of JOSE is composed of "building blocks" called objects, which are the products of an Object Oriented design approach. An object is represented by a *class* however any given class may be possess some or all of the *attributes* of one or more other *base classes*. This property is called *inheritance* and a *class hierarchy* results when one or more *base classes* are *inherited* by another class to derive a new class. This new class is called a *sub-class* and may be defined by all or only some of the attributes of the base classes. The ability to inherit more than one base class is known as *multiple* *inheritance*. These basic Object Oriented design concepts have been used to define the JOSE architecture.

At the executive level of the simulation sits the Environment class. The executive level is the top level in the structure of most programs and it is where much of the "house-keeping" is done. Whilst the primary function of Environment class is to model the characteristics of the environment in which entities exist, it also performs many of the functions of a program executive. An example of its primary function is the determination of Line Of Sight (LOS) between platforms which may affect the ability of sensors to detect other platforms. The executive functions it performs are many and varied. It maintains the list of entities in a scenario and initiates platform/sensor/communications state updates. It also monitors the progress of Monte Carlo simulations and manages data output from these. These represent the principal executive functions performed by Environment class.

The Environment class inherits one class called FatherTime. This contains the simulation time base information and functions to increment simulation time after each iteration. The data and functions associated with this are declared as static because it is not envisaged that there would be more than one instantiation of this class for a simulation. This also ensures that the time base information is readily available to other classes. As can be seen in Figure 1, FatherTime is also inherited by Platform class making simulation time data directly available to this class as well.

The Environment's entity list represents a number of instantiations of the Platform class to which other objects such as sensors may be attached. Platform class sits at the top of a hierarchy of classes which are some of the components which make up a simulation entity. This is depicted in Figure 1 below.

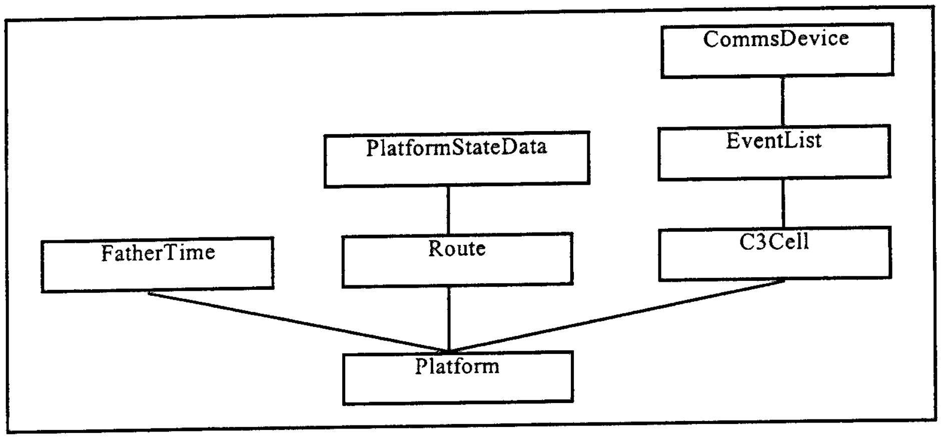

Figure 1. Platform class hierarchy

Platform class manages the motion, if any, of the entity and contains the data which defines the limits of this motion. The data about the current state of the platform (ie.

position, speed, altitude etc.) is maintained in PlatformStateData class. A platform's motion is defined by a route, not surprisingly contained in Route class. This is essentially a list of waypoints which the platform will navigate between. The navigation is performed using great circle computations which assume a curved earth (Reference 5). The motion of a platform is constrained by a number of parameters including a fixed climb and descent rate, and a fixed turn rate; whilst speed can be varied at anytime it is generally defined at each waypoint. The other important property of the Platform class is that the lists of sensors attached to the platform are maintained by it. Currently there is a radar list and an Electronic Support Measures (ESM) list and this can easily be extended to cater for other sensor models.

The role of an entity in a scenario, its relationship to other entities and its current state are managed by C3Cell class. Figure 1 shows that C3Cell is inherited by Platform in the class hierarchy. C3Cell in turn inherits a class called EventList. This class maintains a list of actions which must be performed by the entity at some future time. The CommsDevice class manages receipt and transmission of messages on whatever voice communication links and data links are available to the platform. C3Cell class performs a number of "house keeping" functions. It processes any incoming communications and passes these on for action. In addition, it processes the event list for actions required at the time. Finally it compiles a picture of the world around it based on data from its sensors and from communications links. This picture is called the Tactical Picture. On the basis of information provided by these functions C3Cell will initiate new actions (eg. change course, send a message, scramble fighters etc.) according to certain rules of behaviour which are dependent on the entity's role in the scenario.

The sensors are modelled in solitary classes which can be attached to a platform. Radars, for instance, are managed by a class called RadarClass. The radar sweep is updated according to a schedule (maintained in Schedule class). This defines the radiation pattern of the radar in azimuth so that areas of coverage can be defined. On the basis of this sweep a list of detections for the current iteration is generated. This information is then used to update a track list. A track simply represents a history of detections for a particular target. A track is created when a number of consecutive detections of a target are made and a track is deleted after a period in which the target is not detected.

One other class merits some consideration in this section. This is ScenData class which manages the initialisation of a scenario. This class reads the scenario data files and performs the instantiation and initialisation of platforms and their sensors in the scenario.

2.4 Graphical Display

JOSE was intended to be used for large Monte Carlo simulations, where graphical representation of the scenario in process would be an unnecessary waste of processor time. However, in developing new models and scenarios, a graphical representation of the simulation is a valuable tool to verify intended operations. In addition, if unexpected results occur during a particular Monte Carlo set, individual runs can be replayed with graphics as an aid to understanding the reasons for the outcome.

At present, the graphical display option is only supported on the Amiga platforms. However, as the input files describing the scenario, and the output files generated by a simulation are consistent across all platform types, any run executed on the RS6000 or SG platforms can be replayed graphically on an Amiga.

Figure 2 shows a black and white image of the colour graphical display implemented on the Amiga computer. The display shows icons or symbols representing all of the active platforms in the scenario, coloured to represent either red or blue team participants (lighter icons are red platforms, darker icons are blue platforms in figure 2). The sweeps of radars are drawn at each update, leading to a clear indication of each radar's schedule. The identifier attached to a platform's icon becomes illuminated (light icon text as opposed to dark icon text in figure 2) when the blue team's commander has a currently active track corresponding to it. The flight path of the platforms is illustrated by a track line drawn as the platform moves and this also becomes illuminated (ie. lighter dots in figure 2) when the blue commander has a track for this platform.

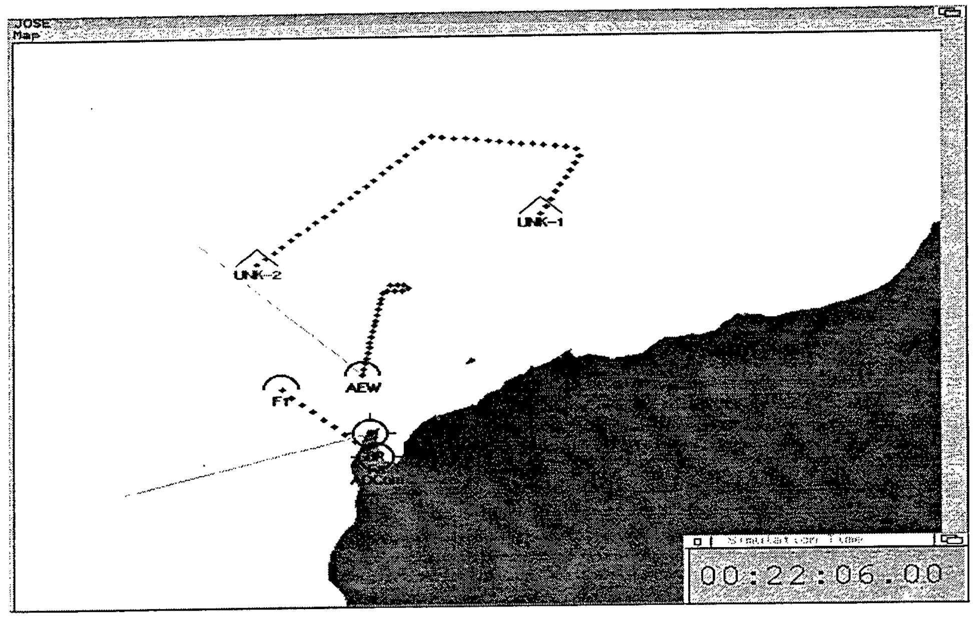

Figure 2. JOSE graphical display on Amiga computer.

2.5 Plotting Program

JOSE is used for Monte Carlo simulations which may consist of thousands of runs, and produces an output file detailing the outcome of each, including information on the nature of the threat and the time and position of intercepts. In this form, this information is not easily interpreted by operations analysts, consequently a better method of visualising the outcome of a Monte Carlo simulation was needed. A purpose-specific plotting utility was developed to aid in this task.

The plotting utility had several requirements:

- To provide a plan view of the area of interest, showing the positions of both the blue and red bases;
- To show where red team members were intercepted by blue fighters;
- To identify which of the blue team's fighters achieved an intercept, thus indicating whether a previous attempt by another fighter failed;
- To indicate when an intercept occurred after a red fighter had reached its weapons release point;
- To display the number of red fighters that achieved weapons release, and the number that were never intercepted; and
- To produce the plot in both encapsulated and printer-ready postscript in colour or black and white.

The usage and output of the resulting "PlotIt" program is described in Section 4.3.

3. Software Implementation Issues

The programming language used was C++ employing the Object Oriented features of this language. Since the project was anticipated to involve a process of continual refinement with input originating from many and varied people, it was decided to develop the software architecture using a Computer Aided Software Engineering (CASE) tool. This allowed classes to be clearly defined and documented before any code was written. During the design process, operations research experts were continually consulted, and the documentation describing how elements of the simulation were intended to be implemented were open to criticism.

The limited availability of computing resources also governed a design decision : the software was to be, as far as possible, platform independent. This was to allow development to take place on existing Amiga desktop systems, with simple graphical representation of the scenario as it played, and also to allow compute-intensive Monte Carlo runs to be hosted on higher-performance platforms such as Silicon Graphics (SG)™ and IBM RISC (RS6000)™ compute servers.

Having multiple programmers work on a single software design can lead to significant administration difficulties. In order to minimise these, the Revision Control System (RCS) package was utilised. This software provides several benefits :

- It acts as a unified repository for all of the application source code, which is then accessible from other computers on the network;
- It ensures that only one programmer can modify any particular file at a time, after 'checking-out' the file for the purpose of writing to it;
- It ensures that source code that has been updated and published ('checked-in') is recompiled by all programmers, so that everyone is using the same (most up to date) versions of the software possible;
- It keeps all revisions of the source files, allowing separate files to restored to previous versions, or a complete build to be made using the version of source code that was valid on any particular date.

4. Using JOSE

The process of using JOSE involves a number of steps. First the user must define a scenario and construct the associated files. Then JOSE is run using these files and output is generated which describes the results of these runs. The output can then be processed with the use of a purpose-specific plotting program. This procedure is described in greater detail in the following sections.

4.1 Defining a Scenario

As has been mentioned previously, JOSE does not have a GUI for scenario definition. Consequently a text editor is required to construct a scenario file and any accompanying radar data files. The structure of these files and the data input statements available to the user are described in detail in the Appendix.

Depending on the complexity of the scenario there may be significant data requirements. The user must be able to place all the entities in a scenario relative to an Earth referenced coordinate system (latitude, longitude, altitude). For each platform the user must be able to specify parameters such as speed(s), climb rate, descent rate, turn rate, radius of action (for fighters). Routes for these platforms may need to be specified either randomly generated or as a list of waypoints. The command hierarchy, if any exists, must be specified and the parameters defining communication links (range, voice delay, network cycle times, etc.) are required. If radars are associated with platforms then data such as azimuthal scan limits, scan rate and detection range(s) are necessary to define this sensor.

Once all this data has been entered into the data files the user can choose to run the scenario singly or as a series of runs in a Monte Carlo simulation. This also is defined by the user in the scenario file.

4.2 Running JOSE

The user must run JOSE from the command line of a shell on the host computer. The format of the command is shown below:

JOSE [options] [scenario file name]

There are a number of options available to the user. Each one must be preceded by a "-" and if there are multiple options required they can be listed one after another. The options available are the following:

d	: Writes messages to the shell when important events occur (eg. intercepts, new tracks etc.).
v	: Verbose mode. This is used for debugging and writes large amounts of data to the shell.
g	: Used to run JOSE without graphics on the Amiga computer.
es	: Save random numbers (to a file called Rands.dat) for replaying particular Monte Carlo runs.
el	: Load random numbers for replaying a Monte Carlo run.

The scenario file name is shown in square brackets because in a sense it is optional as well. If it is omitted then, by default, JOSE will look for a file called Scenario.dat. If this is not found JOSE will terminate execution. Good scenario file management dictates that the user should not give the default name to all scenario files. As JOSE loads the scenario file it checks the data to ensure it is complete and if a problem is encountered an error message is written to the shell indicating where the problem occurred. An example of how JOSE it typically evoked is shown below:

JOSE -d -es Myscenario.dat

4.3 Model Output and Typical Results

The result of running a Monte Carlo simulation is the generation of text files containing the information of relevance to the scenario. Since requirements can differ depending on the purpose of the simulation, JOSE can be re-configured to output more or less information about each run.

Information required for the PlotIt program is output to a file *IntPtsxxx.dat,* where *xxx* is an identifier given in the scenario definition file.

The PlotIt command line is as follows :

plotit [options] [IntPts file name]

where *[options]* can be any number of the following, in any order :

from <fn>	: Specify *<fn>* as the input file. If this option is not present, a list of all options will be shown and no plot produced.
eps	: Encapsulated postscript (default not encapsulated)
blank	: Plot region of interest only, with no intercept data
to <fn | path>	: Specify output file name or directory. If this is not specified, the current directory will be used by default, and the file name will have the same prefix as the input file, but use either ".eps" or ".ps" as the extension (eg IntPts0.dat -> IntPts0.ps).
bw	: Specify black & white rendering (colour is default)
gbr	: Draw ground-based radar coverage areas
nonums :	: Don't overwrite Interceptor numbers on intercept points
plotfile <fn>	: Draw an additional plot on the same axes, using the data specified in *<fn>*. This is typically used for plotting an AEW aircraft's flight path.

The following plot shows the output from an example run where a single threat emanating from an aircraft carrier in the Indian Ocean attacks a base on mainland Australia. This is a purely illustrative example and does not reflect the nature of analyses done for Project AIR 5077.

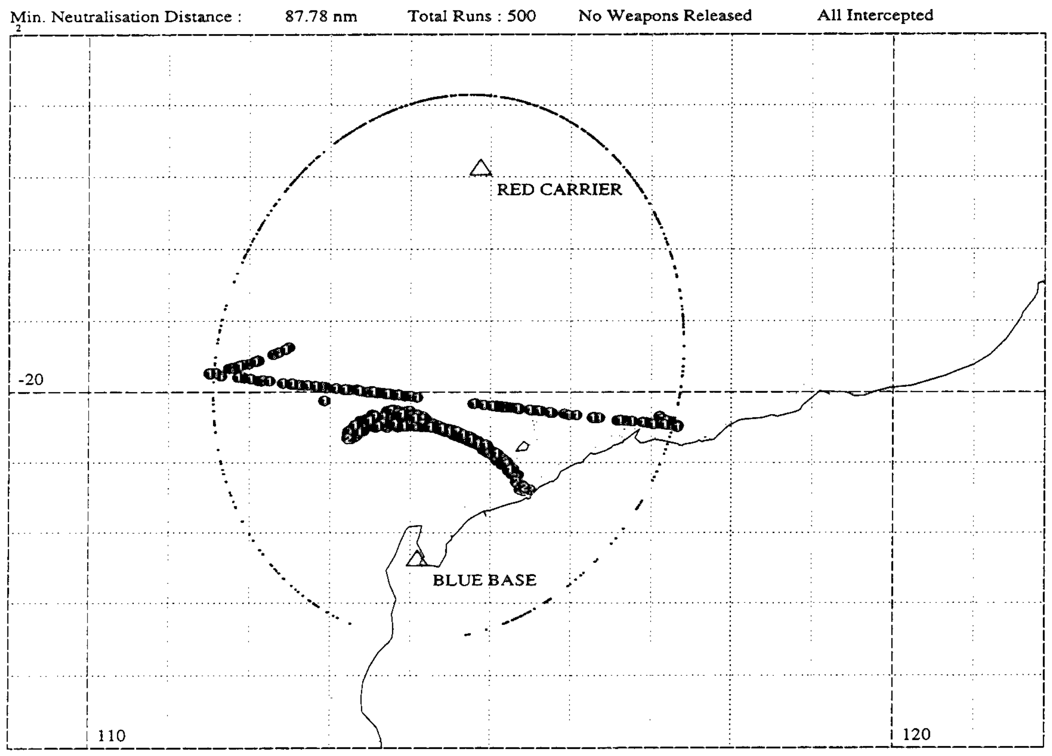

Figure 3. The plot produced by PlotIt for the example run.

Legend :

❶	The position of an intercept that occurred before the intruder reached the point of weapons release. The number in the circle refers to which fighter made the intercept - ie '1' implies the first fighter launched in the scenario.
②	The position of an intercept that occurred after weapons release.
----------	The track followed by an intruder that achieved weapons release.

This simple example shows two main groups of intercepts : a curved group that is closest to the blue base, and a straight-line group further away.

The intercepts along the straight line refer to threats that were detected soon after they took off from the red base, and had intercept points that were calculated to be inside the radius of action of the blue fighters. The curved group represents a different situation where the red fighters` initial flight direction was such that they could not be intercepted, and would fly outside the blue force's radar coverage; however they later turned towards the blue base, and were detected and intercepted. In this example, all of the red fighters were intercepted, and none reached their weapons release point. The elliptical plot of points shows the distribution of turning points for the red fighters, which were given a */randellipse* flight plan (detailed in Appendix 1). The line of information at the top of the plot gives a summary of the simulation, describing numerically the closest distance a red aircraft came to reaching its weapons release point, how many Monte Carlo runs are represented, and details of the number of red fighters that were able to release weapons and return home.

5. Future Development

While JOSE has proved itself of great use to the project so far, it is foreseen that the need for simulation capabilities will increase in the future. Following is an outline of the current plans to extend our capabilities to cater for the anticipated long-term needs of the project and to support the operational capability when it is in place.

5.1 More Detailed Physical Models

JOSE has been designed so that the models of individual systems are clearly differentiated from the workings of the simulation itself. As a result, refining a model or adding a new sensor type to the system is readily achievable. As more information becomes available for particular systems, or as the need to model them more accurately becomes clear, the simulation can be speedily updated to reflect the new requirements.

In the immediate future, modelling effort will be concentrated in the following areas :

- More accurate radar models, using algorithms supplied by Microwave Radar Division (MRD);
- A new ESM capability
- More refined communication link simulation, incorporating Link-4 (Reference 6), Link-11 and Link-16 ; and
- More accurate and detailed platform models and behaviours.

5.2 GUI Development

At present, scenarios are constructed using the text-file protocols described in Section 4.1. While this gives a powerful and flexible way of defining the scenario, it can be somewhat long-winded and awkward. It is proposed to augment the scenario file concept with a graphical scenario editor, in order to make JOSE easier to use.

The scenario editor will targeted at providing the following capabilities :

- The use of libraries of ready-to-use airframes, sensors and communication equipment, so that any particular element need only be defined once and can be shared amongst scenarios (eg an F/A-18 airframe);
- The definition of platforms as a collection of sensors and communications devices attached to a carrier, such as an airframe;
- The use of libraries of pre-defined and user-defined platforms, which in the case of an aircraft, would consist of an airframe accompanied by the relevant sensor suite; and
- The generation of a scenario by instancing one or more platforms from the library, giving initial conditions and tasking where necessary.

5.3 Tactics Development Environment (TDE)

It has been determined that a tool such as JOSE would be useful as an aid in developing and demonstrating the tactical deployment of assets in an air defence role. For this purpose, the Tactical Development Environment (TDE) has been proposed.

The requirements of the TDE differ from that of JOSE in that the TDE needs a "man-in-the-loop" - ie a human to play the role of the Air Defence Commander or a Mission Commander on an AEW. The player will be presented with a graphical representation of the scenario similar to what would be seen on a radar operator's display, from where he can issue orders to the platforms under his control using the communications channels and links as defined by the scenario.

Since large Monte Carlo runs are not applicable to this process, the computational requirements of the TDE are not as severe as for JOSE. It is envisaged that the TDE could be ported to run under Microsoft Windows™, so that ADF personnel could utilise existing computing facilities to run the simulation. It is intended to make platform libraries and scenarios compatible between both applications, and the possibility will exist to feed-back tactics refined by operators using the TDE into JOSE so that they may be tested in a Monte Carlo analysis.

5.4 Human Performance Model

The time required for the simulated human operators to act on information presented to them is currently modelled as a fixed-length delay depending on what procedures would have to be followed. While this is better than assuming a perfect operator, this method does not take into account important parameters such as operator experience, workload and fatigue. Human Factors scientists will be investigating these issues in depth for the proposed AEW systems, and will feed into the simulation more realistic models for operator performance.

5.5 Battle Model

At present, JOSE considers an intercept complete if a fighter is brought to the point where it can engage a threat. To enable a more thorough comparison of different AEW platforms, radars and tactics, it will be necessary to follow the engagement through to completion. This will enable demonstration of the AEW's ability to aid the fighters (via communications links or radio messages) while they are attempting to neutralise a threat.

For this purpose, the concept of a Battle Model has been introduced. While this will use much of the code developed for JOSE, it will also have to model to some extent all of the aspects involved in air-to-air combat, including aircraft and weapons systems performance, and pilot decision making. Since these aspects of the simulation are far from trivial, it is anticipated that the relevant parts of JOSE will be extracted and incorporated into a new environment better able to handle the modelling of a complex reasoning system.

Considerable experience in this field exists within Air Operations Division (AOD) due to the development of the Smart Whole Air Mission Model (SWARMM) system (Reference 7) which uses the Distributed Multi-Agent Reasoning System (dMARS)™ language to model fighter engagements. dMARS is an agent-oriented development environment for building complex, real-time distributed simulations. It can incorporate both reactive and goal-directed behaviour, and is well suited to implementing well-defined tactics and rules whose effectiveness is dependant on many external influences.

It is envisaged that AOD's experience in this area can be capitalised on by using SWARMM as the basis of the battle model. An added advantage would be the ability to better model the performance of other human operators in the simulation, as dMARS is designed to handle such tasks.

6. Conclusion

JOSE has been developed in AOD for the purpose of performing operational effectiveness studies in support of the acquisition process for an AEW&C capability. A modern approach to software development was used employing object oriented programming techniques. JOSE can be used to model an AEW&C, its operating environment and other ADF assets that may be involved in joint operations. It has been successfully used to examine operational effectiveness issues of various candidate AEW&C solutions.

Development of JOSE will continue to meet the anticipated needs of the acquisition process and to support the AEW&C when it becomes operational in the ADF. For the acquisition process development is aimed at providing more extensive and detailed models enabling more fine grained analysis to be performed. Other developments such as the TDE are targeted more at the operational aspects such as tactics and operational procedures. At the heart of all these developments will be JOSE or subset of it (ie. radar model, platform model etc.).

References

1. *DSTO Support to Project AIR 5077 Airborne Early Warning & Control Project Definition Study: Part2,* November 1995.

2. *Supplement to Report DSTO Support to Project AIR 5077 Airborne Early Warning & Control Project Definition Study: Part2,* November 1995.

3. *Understanding Link-11, A Guidebook for Operators, Technicians, and Net Managers,* Logicon Inc., California, USA, 1990.

4. *Understanding Link-16, A Guidebook for New Users,* Logicon Inc., California, USA, 1994.

5. Siouris, G. M. *Aerospace Avionics Systems, A Modern Synthesis,* Academic Press. Inc., California, USA, 1993.

6. *Understanding Link-4A, A Guidebook for Operators, Technicians, and Net Managers,* Logicon Inc., California, USA, 1993.

7. Appla, D. and Steuart, S. *Operational Concept Document for the Smart Whole Air Mission Model (SWARMM),* DSTO General Document, (in preparation).

Appendix 1: Scenario definition

The primary input file for JOSE is the scenario file. By default JOSE will look for a file called Scenario.dat unless the scenario file name is specified at the command line. The format of this file is quite simple and consists of either comments (preceded by a "#") or an actual input statement. All input statements begin with a "/" followed by some descriptor which indicates the nature of the data. The intention of this notation is to make the scenario file as readable as possible and the use of comments is intended to assist this.

The scenario file does have a structure or order for data entry. The first part of the file contains general simulation parameters. This is then followed by the definitions of the entities in the scenario. These are defined starting from the entity at the highest point in the command hierarchy (usually the Air Defence Commander) and the down to the lower levels. One side (ie. blue or red) is defined first and then the other.

A sample scenario is shown below and definitions of the data input statements currently available to the user is presented after that.

```
#####################
# Simulation parameters. #
#####################

# Simulation time data.
/timestep 1.0 0.0

# Data output definitions
/dataout 1000.0 0

# Monte Carlo simulation parameters.
/montecarlo 1000 20.0 30.0 0

# Universal comms parameters for blue net (ie Link 11 net).
/definebluenet 500.0 5.0 7.0 3.0

#####################
# Commander definition. #
#####################

# Commander Platform definition
/platform ADcomm Blueteam Groundbased 0.0 0.0 0.0 0.0

# Commander static position
```

```
/waypoint -22.4 114.09 0.0 StaticPoint

# Define this entity as the commander
/c3cell ADcommander 1 0 OnTask 0.0 360.0 0.0

# Initialise this entity as member of blue net.
/bluecomms

##################
# AEW&C definition.  #
##################

/platform AEW Blueteam Airborne 350.0 2.0 5000.0 5000.0
/c3cell AEW 2 1 OnTask 1000.0 300.0 0.0
/bluecomms

# Racetrack orbit with randomly generated starting point.
/randorbit -19.50 114.0 -19.50 114.5 30000.0 clockwise 1 2 300

# Alternate station to fall back to.
/altstation -21.0 114.0 30000.0 AntiClockwise 30.0 20.0 RaceTrack

# AEW radar definition.
/radar 240radar.rad PlatformReferenced 0.0 240radar.sch

###############
# GBR definition.  #
###############

/platform GBR Blueteam Groundbased 0.0 0.0 0.0 0.0
/waypoint -22.0 114 0.0 StaticPoint
/c3cell ReportingPoint 3 1 OnTask 0.0 0.0 0.0
/bluecomms
/radar GBradar.rad EarthReferenced 0.0 GBradar.sch
```

```
########################
# Blue interceptor definitions. #
########################

/platform F1    Blueteam Airborne 670.0 1.0 20000.0 10000.0
/waypoint -22.4 114.09 0.0 StaticPoint
/c3cell Fighter 4 1 Alert3 350.0 0.0 0.0
# Fighters use radio communications
/comms 500.0 5.0

/platform F2    Blueteam Airborne 670.0 1.0 20000.0 10000.0
/waypoint -22.4 114.09 0.0 StaticPoint
/c3cell Fighter 5 1 Alert15 350.0 0.0 0.0
/comms 500.0 5.0

/platform F3    Blueteam Airborne 670.0 1.0 20000.0 10000.0
/waypoint -22.4 114.09 0.0 StaticPoint
/c3cell Fighter 6 1 Alert30 350.0 0.0 0.0
/comms 500.0 5.0

/platform F4    Blueteam Airborne 670.0 1.0 20000.0 10000.0
/waypoint -22.4 114.09 0.0 StaticPoint
/c3cell Fighter 7 1 Alert30 350.0 0.0 0.0
/comms 500.0 5.0

#################
# Threat definitions.  #
#################

# First threat.
/platform UNK-1 Redteam  Airborne 500.0 1.0 10000.0 10000.0
/c3cell Fighter 8 0 FollowingWaypoints 1000.0 0.0 0.0
/comms 500.0 5.0
/randellipse -16.90 114.87 -22.4 114.09 800.0 30.0 500.0 500.0 500.0 50000.0 120.0 100.0
10.0 Yes -30.0 100 22222 300 500.0 550.0 500.0 450.0 450.0 500.0 550.0 400.0 0.0

# Second nasty.
/platform UNK-2 Redteam  Airborne 500.0 1.0 10000.0 10000.0
/c3cell Fighter 9 0 FollowingWaypoints 1000.0 0.0 0.0
/comms 500.0 5.0
/randellipse -16.90 114.87 -22.4 114.09 800.0 30.0 500.0 500.0 500.0 50000.0 120.0 100.0
10.0 Yes 30.0 100 222 300 500.0 550.0 500.0 450.0 450.0 500.0 550.0 400.0 60.0
```

The data input statements available to the user for defining a scenario are described in detail below:

General Simulation Data

/timestep (time step) (stop time)

This statement defines the simulation time step (in seconds) and a stop time if required (in seconds). If the stop time is set to zero then the simulation will continue indefinitely unless stopped by the user or some other termination criteria.

/dataout (output frequency) (simulation run no.)

This statement defines an output frequency (in seconds) relative to the simulation time base for data to be output to files. The simulation run number (0-999) is appended to output files to assist in identifying runs.

/montecarlo (runs required) (interception range) (neutralisation time) (replay run no.)

This statement is used for defining Monte Carlo simulation parameters. A Monte Carlo simulation may consist of many individual runs of the model. The first parameter defines how many are required for a simulation. The next two parameters relate to the criteria currently used to end one run and start another. When a blue fighter comes within interception range (in nautical miles (nm)) a neutralisation time is assumed (in seconds) and the threat is defeated. If no other threats remain a new run is initiated. The replay run number is used to replay a particular Monte Carlo run. This can take a value from 0 to (runs required) where 0 means a replay is not required. The run numbers are obtained by looking at the output files. Generally we are only interested in replaying one particular run so the (runs required) must be set to 1.

/definebluenet (range) (voice delay) (net cycle time) (No. of participating units)

This statement defines the parameters for the blue data link network. The first parameter is the range limitation for the participants (in nm). If a voice message is sent then voice delay is the time delay before it is received (in seconds). The net cycle time (in seconds) is the time required to service all participants in a network if we are defining a Link 11 data link. For other data links this may be set to zero. The number of participating units is used to divide the net cycle time into equal shares defining the time interval over which they have control of the network and are able to transmit data.

Entity Definition Data

/platform (textual identifier) (team) (platform type) (velocity) (turn rate) (climb rate) (descent rate)

The */platform* statement usually heads a block of statements defining the attributes of an entity (ie. route, sensors etc.). Thus when the next */platform* statement appears, definition of the previous entity ceases and definition of a new entity begins. The first parameter is a textual identifier (max. 10 characters) which is used for the graphical display. The next parameter is the platform team (ie Blueteam or Redteam) followed by the platform type (ie Airborne, Surface or GroundBased). The velocity (in knots), turn rate (in degrees/second), climb rate and descent rate (in feet/minute) are the final parameters.

/waypoint (latitude) (longitude) (altitude) (waypoint type)

A platform must have at least one waypoint to position it. A waypoint definition starts with the latitude and longitude (in degrees) followed by the altitude (in feet). The last parameter is the waypoint type (StartPattern, StartRoute, WayPoint, EndPattern, EndRoute, or StaticPoint). The route a platform will follow is composed of at least two waypoints (StartPattern and EndPattern or StartRoute and EndRoute) with as many intermediate waypoint (Waypoint) as required. The difference between the StartPattern/EndPattern and StartRoute/EndRoute is that a platform will cycle around the first one indefinitely whereas in the second case the platform will cease motion at the EndRoute waypoint. The StaticPoint is used to define the position of platforms which are motionless however this does not preclude them from moving if it is appropriate for that platform (eg. they may be vectored).

/waypointvel (latitude) (longitude) (altitude) (velocity) (waypoint type)

The difference between this waypoint definition and the previous one is that a velocity can be specified (in knots) which the platform will take upon reaching this waypoint.

/randorbit (latitude1) (longitude1) (latitude2) (longitude2) (altitude) (orbit direction) (seed1) (seed2) (seed3)

For Monte Carlo simulations the initial position along a route or some other aspect of the route may need to be generated randomly. The random orbit function is used to define a racetrack orbit where the initial position in that orbit is randomly determined. The ends of the orbit are defined by the (latitude1,longitude1) and (latitude2,longitude2) coordinate pairs (in degrees) at the specified altitude (in feet) and orbit direction (Clockwise/AntiClockwise). The three seed values (0 to 65535) are used by a random number generator in determining the platform's starting position.

/altstation (latitude) (longitude) (altitude) (orbit direction) (orbit orientation) (orbit length) (orbit type)

An entity may startup at some station perhaps defined by the */randorbit* statement. It may at some point in a scenario choose to move to an alternative station defined by */altstation*. This station is defined by a latitude and longitude (in degrees) and an altitude (in feet) with a certain orbit direction (Clockwise/AntiClockwise). The orientation and length inputs apply only to the racetrack orbit type. The orbit orientation is defined with respect to the North axis (in degrees) and the length is the distance between the turning points at the end of the racetrack (in nm). The two orbit types available are Racetrack and Circular.

/randellipse (base latitude) (base longitude) (target latitude) (target longitude) (range) (no overfly radius from target) (inbound altitude) (ingress altitude) (egress altitude) (outbound altitude) (dash in distance) (dash out distance) (weapons release range) (using random departure or fixed) (fixed departure angle) (seed1) (seed2) (seed3) (base climb out velocity) (outbound velocity) (descent to ingress velocity) (ingress velocity) (egress velocity) (post egress climb velocity) (outbound velocity) (base descent velocity) (departure time)

The */randellipse* function is generally used to describe the route taken by a red strike aircraft. It describes a HI-LO-LO-HI profile emanating from the platform base with a target at some point. This profile is constrained by an ellipse with the base and the target as the foci and the size determined by the range of the platform and distance between base and target. The platform takes off, climbs and heads toward the elliptical boundary. At the boundary it turns toward the target. At some distance along this track the platform descends to an ingress altitude and at the weapons release distance from the target it turns toward home base. It continues at the specified egress altitude for a specified distance. It then climbs to altitude until finally descending back to its base. The angle at which the platform departs from its base can be randomly generated (for Monte Carlo simulations) or defined by the user. A radius around the target can be defined which the platform will not overfly before turning at the elliptical boundary. In addition the departure time with respect to the simulation time can be defined so that strikers can be staggered in time.

The input parameters begin with the latitude and longitude of the base and target (in degrees). Then the platform range and radius around the target which must not be overflown by the red strike aircraft whilst inbound is specified (both in nm). The inbound, ingress, egress and outbound altitudes follow (in feet). The next parameters are the dash in to target distance, dash away from target distance and weapons release range (in nm). Next the user specifies whether departure angle is randomly generated or not (Yes/No). This is followed by the fixed departure angle (-180.0 degrees to 180 degrees with 0 degrees being directly at target) and the random number seeds (0 to 65535) for random departure angle generation. The platform velocities for all eight segments of the profile must be specified (in knots). Finally the departure time (in seconds) must be specified.

/randtgts (latitude1) (longitude1) (latitude2) (longitude2) (seed1) (seed2) (seed3) (rate of effort) (speed) (altitude) (target life) (No. of targets to be generated) (min. no. of targets to be generated) (error bound)

The */randtgts* function has a specific purpose related to leakage rate computations for an airborne sensor such as an AEW. It is used to generate waves of targets so that the performance of the airborne sensor on some barrier patrol can be measured (ie. by leakage rate). The targets are generated along the line between points 1 and 2 in a uniformly distributed manner. The targets fly away from this line in a direction which is equal to the line direction (point 1 to point 2) plus 90 degrees. They fly at the specified speed and direction for a period equal to the target life and then disappear. The targets are generated at the rate specified by the rate of effort. There is a minimum number of targets that must be generated and a maximum number to generate. The error bound is computed for the leakage rate and if the required error bound is achieved before the maximum number of target has been generated, the simulation will terminate.

The first four parameters are the latitude and longitude for the two end points of the line (in degrees). Next are the random number seeds (0..65535) used in generating the targets. These are followed by the rate of effort (targets/hour), the speed (knots), altitude (feet) and target life (hours). The maximum number of targets to be generated (0..32767) is specified followed by a minimum number (0..32767) which must be generated before the simulation can terminate. Finally an error bound for the leakage rate must be specified (eg. 0.001 means the leakage rate must be accurate to +-0.001).

/c3cell (type) (command id) (commander id) (initial state) (radius of action) (C2 delay) (fighter acceleration delay)

The c3cell parameters define the entity's place in the command hierarchy and the behaviour that it exhibits. The first parameter defines what type of entity it is (ADcommander,AEW,Fighter,ReportingPoint) and its rules of behaviour are determined by this. The next parameter is a command id for this entity (0 to 65535) where 0 implies it is at the lowest level in the hierarchy (ie. many entities can have a command id of 0) and any non zero id must be unique. This is followed by the id of this entity's commander which will have been defined previously in the scenario file. Next comes the initial state for which there are a number of possibilities. The states of Alert3, Alert15 and Alert30 apply to fighter aircraft on ground alert where the 3, 15 and 30 refer to the time required to become airborne. The Alerted state implies that an entity has ended its alert state and is about to become airborne or is airborne. The OnTask state means that the entity is engaged in some task (eg. an AEW on patrol). The last state called FollowingWaypoints applies to "dumb" entities which do nothing but follow their predefined route. The next parameter is the radius of action (in nm) which is used to compute whether a fighter can intercept a certain target. Following this is the C2 delay (in seconds) which is the time taken by an entity to assess a situation and decide what action it will take. Finally the acceleration delay (in seconds) is used to compensate for the intercept calculation which assumes the same speed (ie. the intercept speed) for the fighter throughout the intercept profile. In fact the speed during climb out will be less than the straight and level intercept speed. To compensate for this a takeoff delay is imposed and this must be precomputed by the user.

/bluecomms

This statement initialises the current entity as a participating unit in the blue network defined in the general simulation data at the top of the scenario file.

/comms (range) (voice delay)

This statement defines this entity as possessing a radio for voice communications with other entities. The first parameter is the transmission range (in nm) for the communications while the last parameter defines the delay before a transmission is received.

/radar (radar data file name) (azimuthal reference axes) (initial azimuth) (radar schedule file name)

This statement associates a radar with a entity. This first parameter is the name of the file containing the parameters for the radar. The next parameter defines the axes to which the sweep azimuth is referenced to (EarthReferenced/PlatformReferenced). Earth referenced means the sweep will be stabilised relative to the earth and will be unaffected by platform motion. Platform referenced means that sweep is relative to the platform so that platform turns will affect the sweep relative to the earth. The initial azimuth parameter (in degrees) then depends on the previous parameter (for EarthReferenced zero is North and for PlatformReferenced zero points to the front of the platform). Finally the file name for the radar schedule is specified.

Radar Definition Files

To define a radar two files are required. The first is the radar data file which contains the parameters describing what type of radar it is. The second file contains the radar schedule. A schedule defines the radiation pattern in azimuth for the radar. Samples of these two files are given below:

```
# Radar data file for AEW.
/radardata 0.0 360.0 18.0 Clockwise Rotational PhasedArray240 Off
```

```
# AEW radar schedule data file.
/scheddata 30.0 240.0
/scheddata 150.0 0.0
/scheddata 210.0 240.0
/scheddata 330.0 0.0
```

Thus a radar data file will contain one statement of the form shown below:

/radardata (start azimuth) (end azimuth) (sweep rate) (sweep direction) (radar type) (radar technology) (On/Off in a turn)

The first parameters in the */radardata* statement are the start and end limits (in degrees) for the azimuthal scan. Next is sweep rate (in degrees/second) and direction of sweep (Clockwise/AntiClockwise) for the radar. The radar type parameter refers to how the radar sweeps in azimuth (Rotational/Oscillating). Rotational means it scans in the one direction all the time whereas oscillating means that it scans back and forwards between the azimuth limits. The next parameter is appropriate to airborne radars (Rotordome/PhasedArray240/AirToAir). The first two options refer to AEW radars while the last is for fighter radars. The last parameter allows the user to specify whether the radar switches off during a turn (On/Off).

A radar schedule file may contain many statements however they are all of the same type. The format of this statement is shown below:

/scheddata (start azimuth) (reference value)

This statement is referred to as a schedule step and there is no limit to the number of these that can appear in a schedule file. One statement can be used to define the whole schedule or a number can be used to define each part of the whole schedule. The first parameter is the azimuth (in degrees) that the schedule step takes effect from and the second parameter is in fact the detection range (in nm) for the radar in that schedule step. Thus if only one */scheddata* statement is used then that detection range applies for the whole of the azimuth scan for that radar. Using multiple statements allows the azimuthal scan range to be divided into sectors with different detection ranges as shown in the example above. This can extend over more than one sweep of the radar so that a schedule can vary from one sweep to the next.

JOSE - Joint Operations Simulation Environment

M. Iob and D.A. Craven

AUSTRALIA

1. DEFENCE ORGANISATION

a. Task Sponsor Director General Aerospace Development

b. S&T Program

Chief Defence Scientist ⎫
FAS Science Policy ⎬ shared copy
AS Science Corporate Management ⎭
Director General Science Policy Development
Counsellor Defence Science, London (Doc Data Sheet)
Counsellor Defence Science, Washington (Doc Data Sheet)
Scientific Adviser to MRDC Thailand (Doc Data Sheet)
Director General Scientific Advisers and Trials/Scientific Adviser Policy and Command (shared copy)
Navy Scientific Adviser (Doc Data Sheet and distribution list only)
Scientific Adviser - Army (Doc Data Sheet and distribution list only)
Air Force Scientific Adviser
Director Trials

Aeronautical and Maritime Research Laboratory
Director

Chief of Air Operations Division
RL-AV
RL-AP
Head of Discipline - HOAP
Task Manager - HAPA
M. Iob (2 copies)
D.A. Craven (3 copies)
I. Lloyd
R. Brown
L. Mockridge
B. Hanlon
G. Lawrie
T. O'Connor
P. Blanchonette

DSTO Library
Library Fishermens Bend
Library Maribyrnong
Library Salisbury (2 copies)
Australian Archives
Library, MOD, Pyrmont (Doc Data sheet only)

c. **Capability Development Division**
Director General Maritime Development (Doc Data Sheet only)
Director General Land Development (Doc Data Sheet only)

d. **Navy**
SO (SCIENCE) Director of Naval Warfare, Maritime Headquarters Annexe

e. **Army**
ABCA Office, G-1-34, Russell Offices, Canberra (4 copies)

g. **Intelligence Program**
Defence Intelligence Organisation
Library, Defence Signals Directorate (Doc Data Sheet only)

i. **Corporate Support Program (libraries)**
OIC TRS, Defence Regional Library, Canberra
Officer in Charge, Document Exchange Centre (DEC), 1 copy
*US Defence Technical Information Centre, 2 copies
*UK Defence Research Information Center, 2 copies
*Canada Defence Scientific Information Service, 1 copy
*NZ Defence Information Centre, 1 copy
National Library of Australia, 1 copy

2. **UNIVERSITIES AND COLLEGES**

Australian Defence Force Academy
  Library
  Head of Aerospace and Mechanical Engineering
Deakin University, Serials Section (M list), Deakin University Library, Geelong, 3217
Senior Librarian, Hargrave Library, Monash University
Librarian, Flinders University

3. **OTHER ORGANISATIONS**

NASA (Canberra)
AGPS

OUTSIDE AUSTRALIA

4. **ABSTRACTING AND INFORMATION ORGANISATIONS**
INSPEC: Acquisitions Section Institution of Electrical Engineers
Library, Chemical Abstracts Reference Service
Engineering Societies Library, US
Materials Information, Cambridge Scientific Abstracts, US
Documents Librarian, The Center for Research Libraries, US

5. **INFORMATION EXCHANGE AGREEMENT PARTNERS**
Acquisitions Unit, Science Reference and Information Service, UK
Library - Exchange Desk, National Institute of Standards and Technology, US
National Aerospace Laboratory, Japan
National Aerospace Laboratory, Netherlands

Page classification: UNCLASSIFIED

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION DOCUMENT CONTROL DATA	1. PRIVACY MARKING/CAVEAT (OF DOCUMENT)
2. TITLE JOSE - Joint Operations Simulation Environment	3. SECURITY CLASSIFICATION (FOR UNCLASSIFIED REPORTS THAT ARE LIMITED RELEASE USE (L) NEXT TO DOCUMENT CLASSIFICATION) Document (U) Title (U) Abstract (U)
4. AUTHOR(S) M. Iob and D.A. Craven	5. CORPORATE AUTHOR Aeronautical and Maritime Research Laboratory PO Box 4331 Melbourne Vic 3001

6a. DSTO NUMBER DSTO-TR-0533	6b. AR NUMBER AR-010-226	6c. TYPE OF REPORT Technical Report	7. DOCUMENT DATE May 1997	
8. FILE NUMBER M1/9/164	9. TASK NUMBER ADA 94/245	10. TASK SPONSOR DGFD (Aerospace)	11. NO. OF PAGES 25	12. NO. OF REFERENCES 7

13. DOWNGRADING/DELIMITING INSTRUCTIONS None	14. RELEASE AUTHORITY Chief, Air Operations Division

15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT

Approved for public release

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE CENTRE, DIS NETWORK OFFICE, DEPT OF DEFENCE, CAMPBELL PARK OFFICES, CANBERRA ACT 2600

16. DELIBERATE ANNOUNCEMENT

No Limitations

17. CASUAL ANNOUNCEMENT Yes

18. DEFTEST DESCRIPTORS

simulation, airborne early warning and control system, command control communications, operations research

19. ABSTRACT

JOSE has been developed for use in operational effectiveness studies performed in support of the acquisition process for Project AIR 5077 Airborne Early Warning and Control (AEW&C). It can be used to simulate an AEW&C and the Australian environment it must operate in. This includes platforms, sensors, C3 structure and other ADF assets that may be involved in joint operations. Development of JOSE will continue to meet the needs of the acquisition process and to support the capability when it enters service.

Page classification: UNCLASSIFIED