A Conceptual Model for the Computer Simulation of Air Traffic Control Radar Surveillance Systems

Reza Adibnia

B.A.Sc. (Electrical Engineering), Sharif University of Technology, 1999

THE THESIS SUBMITTED IN PARTIAL FULFILMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

THE FACULTY OF GRADUATE STUDIES

(ELECTRICAL AND COMPUTER ENGINEERING)

THE UNIVERSITY OF BRITISH COLUMBIA

September 2005

Abstract

The objective of this work is to identify and characterize a fundamental set of abstractions to design a software tool to simulate the behaviour of one or more Air Traffic Control (ATC) Primary Surveillance Radar (PSR) and Secondary Surveillance Radar (SSR) systems.

The Air Traffic Control (ATC) radar data is very important in the provision of ATC services. Air traffic controllers use radar data to assure the separation of aircraft. Radar data contains information about the altitude of an aircraft. It also may have indications of emergency conditions transmitted by means of a transponder from an aircraft to a radar source on the ground. Simulating the generation of these radar data is important for training ATC operators and testing other ATC equipment that uses radar surveillance data as input.

To simulate the generation of radar data, it is required to simulate aircraft, surveillance radar and the environment between aircraft and radar, which may consists of simulation of weather conditions and obstructions. Simulation provides useful means to analyze and study the reaction of ATC radar system against different conditions. In particular, simulation avoids the high cost and often high risk of using real inputs (i.e., flying real aircraft) for the purposes of training and equipment testing

We have also demonstrated the practical application of the concepts presented in this dissertation through the implementation of an ATC radar simulation tool.

Table of Contents

List of Tables

List of Figures

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Dr. Jeff Joyce, for his constant help and kind guidance throughout this work. I will not be able to thank Dr. Joyce, enough, whom I respect and admire tremendously, for his support during my studies. His expertise in the area of Software Engineering and his background in the Air Traffic Control not only made this work possible but also enriched my personal and professional life.

I also would like to thank my kind wife, Setareh Aslani, for her never-ending patience, constant encouragement and support.

1 Introduction

1.1 Introduction

The objective of this work is to identify and characterize a fundamental set of abstractions for the software simulation of a complex physical system – namely, the surveillance of airspace by a network of Air Traffic Control (ATC) radar. The generation of simulated ATC radar surveillance data is important for both the testing of ATC systems and the training of air traffic controllers. Real-time simulation of all the physical phenomena that affect ATC radar surveillance would far exceed any reasonable configuration of computational resources. This dissertation describes a fundamental set of abstractions that may be used in place of more detailed models of physical phenomena for the purpose of generating simulated ATC radar surveillance data.

Using these abstractions, we have implemented a tool that generates simulated ATC radar surveillance data. This software tool, "Radar Source Simulator" (RSS), simulates the behaviour of one or more Air Traffic Control (ATC) Primary Surveillance Radar (PSR) and Secondary Surveillance Radar (SSR) systems. A simulation script specifies the activity in this simulation. This script defines the creation of one or more flights with a variety of characteristics including the trajectory of the flight. Changes to these characteristics may also be scheduled in the simulation script. The behaviour of each simulated radar source is determined by a number of characteristics specified in the simulation script – for example, the location of the radar source and its range. This tool includes a means of modeling obstructions such as tall buildings and mountains that limit radar coverage. Finally, the tool provides a means of simulating the relationship between barometric pressure and altitude contained in SSR radar data.

1.2 The importance of ATC Radar Simulation

The first priority of an air traffic controller is to ensure that aircraft under his or her jurisdiction are adequately separated from other aircraft, terrain and special use airspaces such as weapons testing ranges. The current position and altitude of the aircraft are key inputs for separation decisions.

There are two different basic kinds of position and altitude information: information reported by pilot using radio communication and information obtained by radar. In general, radar surveillance data is considered more accurate than reported data. The accuracy of reported data is dependent on the accuracy of onboard navigation systems and flight conditions. Nevertheless, positional information reported by pilots using radio is not completely effective in dense traffic areas and the use of ground radar is required to obtain accurate aircraft position data in these areas. This is reflected in rules used by air traffic controllers. For example, the lateral separation of two aircraft at the same altitude in the case of having radio communication is 5 nautical miles. This separation may be decreased to 3 nautical miles (subject to other conditions) in the case of having radar surveillance data.

In addition, surveillance radar systems help controllers to identify the aircraft (the name of the flight or whether the aircraft is the enemy or an ally, commercial or military. They also give some information about the status of the flight which might be useful in predicting the flight situation, for example controllers can find out the actual speed of an aircraft from radar data that is useful to predict the time of arrival to the destination – the actual speed might be different from the speed which is measured by onboard systems. Other beneficial outcomes are better flight paths, improved operation efficiency, and the ability to provide navigational assistance to flights.

The radar system is one the significant parts of an ATC system. The radar system includes transmitter-receiver equipment and Plot Extraction Unit, which converts radar replies into a stream of radar returns. This stream of radar returns is sent to a Radar Data Processing System (RDPS) where it is combined with radar returns from other radar sources. From these radar returns and other data, the RDPS produces a stream of tracks,

which may be displayed directly to controllers or used as inputs by other ATC equipment. Figure 1 is simple sketch of an ATC radar system.

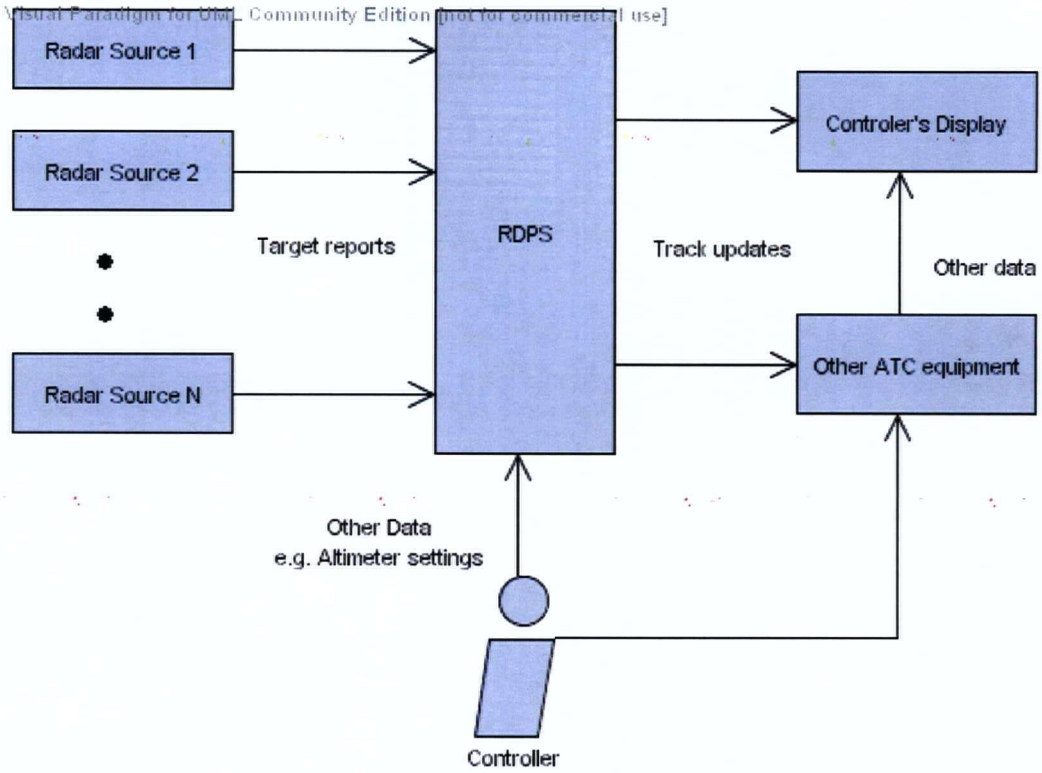

Figure 1: A simple sketch of an ATC system

The development of a RDPS requires a means of generating simulated input for the purpose of testing. In addition to testing for compliance with the functional requirements of the RDPS, the generation of simulated input is required for both performance and safety testing. Simulation is necessary because it is not practical to depend exclusively on real "live" input for the purpose of testing. In addition to the prohibitive cost of flying real aircraft, it would be too dangerous to create some of the conditions that need to be examined in the context of safety testing. For example, having two aircraft flying in close proximity to each other would be unacceptably hazardous. The ability to generate simulated input is also likely to be required to train air traffic controllers. Instructors will prepare a variety of training exercises in which aircraft movements and other conditions, such as the failure of a radar source, are scripted. These scripts are then executed by this simulation tool to generate input for the RDPS in the training facility.

1.3 Computer Simulation of Physical Systems

Computer simulation is re-creation of a physical system with a computer model, which can be processed by a computer program. It is a way of making experimental measurements on a real system by substituting it with a simulated system. These measurements are made on state variables, which define the desirable aspects of the system [1].

Computer simulation has become useful in modeling physical systems to gain insight of system's operation and to predict the behaviour of the system. The benefit is that computer simulation of a system is usually much more feasible, cheaper, faster and safer (while posing less of a risk) than experimenting with real objects in a real environment.

The findings of the Board of Inquiry for the 1996 Ariane 5 rocket disaster speak very clearly to the importance of simulating physical system accurately when these simulations are used as a means of testing critical software. The Board of Inquiry created to investigate this disaster found that the testing of critical software under realistic input data was inadequate. Since this was the first launch of the Ariane 5 class, the only source of realistic input data would have been simulation. Simulations performed after the disaster "have faithfully reproduced the chain of events leading to the failure of the inertial reference system" [2].

Most computer simulations of physical systems fundamentally involve a set of state variables that change value as the passage of time is simulated. A primary consideration in the early stages of developing a simulation model is to determine when these state variables will be updated. Should these state variables be updated once for every minute of simulation time? Or should they be updated once every second for greater fidelity? Alternatively, it may be sufficient to update these state variables only once every hour of simulation time. The answer to this question depends greatly on the kind of physical system being updated? Moreover, it may not be efficient to update every state variable at

the same rate since some variables may change more quickly than others. To answer such questions, we consider three basic categories of simulation models:

Continuous simulation: In a continuous simulation, measurements are made on state variables, which have continuous changes as time advances [1]. Theoretically, the simulation program of the continuous model recalculates all the equations, which are calculating the state variables repeatedly. Since this implementation is inefficient, the equations are solved by a technique in which time advances in small time steps with calculations to update the state variables [3]. In this method there is a trade-off between accuracy and speed, and the determinant of this trade-off is time step size [4].

Discrete-event simulation: In a discrete-event simulation, measurements are made on state variables, which are affected by events [1]. The system states change only at a set of discrete points in time called event time [5]. The simulation is advanced from event time to event time rather than using a continuously advancing clock. In this type of simulation, the simulator maintains a queue of events sorted by time at which they should occur. The simulator reads the queue and triggers new events as each event is processed. No simulated time passes when an event occurs. It passes only between each event. In discrete simulation, it is not necessarily important to execute the simulation in real time but it is more important to be able to access the data produced by the simulation.

Combined simulation: A combined simulation deals with the integration of continuous and discrete-event model elements. On one hand, changes of continuous state variables may cause events to occur, which may affect discrete-event state variables whereas on the other hand, events may change continuous state variables and alter continuous model behaviour [6].

In comparison, a continuous simulation is often less accurate than discrete-event simulation. The reason is that to model a continuous state variable on a computer we need to make its representation discrete and the interval between measurements of the state variable has an impact on precision of system's behaviour.

The implementation of both discrete-event and continuous simulation are easier than the implementation of combined simulation. The key concept in combined simulation, to interface continuous and discrete-event simulation, is that the start and the end of integration process between continuous and discrete parts are events. For example, suppose that a discrete simulation has a continuous process embedded within it. This continuous process will be triggered to begin when some specific event occurs within the discrete model. A continuous simulation process now starts and continuous integration is processed step by step to completion upon the satisfaction of some criterion. It is important that the step size of the integration process be controlled so as to synchronize correctly with events so that events coincide with the end of an integration step. This process requires that some variable-step integration methods are used for the synchronization and this makes combined simulation more complicated than other two types of simulation [3].

The RSS is a combined simulation. It is an integration of the aircraft system whose states have continuous changes (such as the location of the aircraft changes over time) and surveillance radar system whose states have discrete changes (such as the beam position at which radar sends its energy impulse varies discretely).

1.4 Challenges of Radar Simulation

The design and implementation of a tool to generate simulated ATC radar data is complicated by a number of factors which might not be initially apparent to software developers. For example, the altitude information that may be contained in ATC radar data is not simply the height of the simulated aircraft above sea level. Rather, this altitude information is also a function of barometric pressure. Therefore, the simulation model must include functional elements for barometric pressure so that the consequences of variation in barometric pressure can be observed during both testing and training activities. Other subtleties in the generation of simulated ATC radar data may include modeling the consequences of real world phenomena such as the multi-reflection of ATC radar signals that create false targets. Also, transponder errors may make the detection of

the target range difficult. Furthermore, the design and implementation of such a tool should anticipate the special needs of safety testing that might involve the generation of simulated radar data with particular anomalies, such as the incorrect information of an aircraft's altitude.

A part of the RSS is a discrete simulation of continuous phenomena. In this sort of simulation, there is always a trade off between the rate of approximation, taken for the simulation, and the effective cost of simulation (speed and memory). For example, choosing an appropriate time interval to update the location of an airplane is fundamental. Whereas choosing small intervals may improve the simulation, it is expensive to calculate. While choosing large intervals is inexpensive, it may weaken the effectiveness of the simulation. The best solution for this problem is to make decisions in light of the situation. As in our example, depending on the aircraft situation, RSS uses variable time intervals to update the aircraft's location.

1.5 Related work

There have only been few papers published about the computer simulation of radar surveillance. Most of these publications belong to one of the following categories:

1. Radar system simulation
2. Aircraft movement simulation
3. RDPS simulation
4. Tracking system simulation

The article "The Use of Radar Environment Simulation for Operator Training and System Test and Evaluation" [6] was published in the 1990's. It introduces general simulation architecture for radar environment. It suggests that radar environment must provide three major functions: data preparation, scenario modelling and pulse generation. It focuses on functions to provide aircraft movements, signal pulses and signal propagation. At the end,

the paper suggests the future work on propagation effects caused by the surrounding environment.

The article "Computer Modelling of Advanced Radar Techniques and the Advanced Radar Simulator" [8] explains a radar model with particular reference to the importance of ensuring that the model's structure accurately represents the real physical process. The model takes attention to radar transmitter and receiver, signal propagation and attenuation, reflectivity of the target and noise. The model simulates the whole sequence that happens to the signal's wave from transmitting to receiving. But the model does not present some important environmental effects like reflection on signal propagation. It also does not describe any method to simulate movements of the aircraft.

The article "Computer Simulation of Marine Search and Rescue Operations Using Newly Developed Radar Beacon" [9] introduces new attention in radar simulation. In addition to signal propagation, the model takes into account the signal reflection over sea waves and the effect of the earth's curvature in target detection. It also describes necessary conditions for successful detection by radar and a computer algorithm to implement it.

Part of the article "Object-oriented Military Simulation Development and Application" [10] describes the critical factors in radar simulation environment. Without giving any model for simulation, the article briefly explains that a simulation environment must be composed of coordinate systems, motion, energy propagation, physical object interaction and event timing.

The article "A Development Methodology Applied to a Radar System Simulation", [11] published in 1983, introduces a top-level design for radar system simulation. It presents a functional and hardware model architecture for the radar system. In a functional model, the system consists of a:

1. Scenario: Generates the positional data for targets and environment.
2. Sensor: Simulates the radar transmitter, receiver and antenna.

3. Search: Simulates activates of initial target detection.
4. Track: Simulates activities of target detection after target initiation.
5. Resource management: Optimizes the functions of search and track.

The paper also describes a top-level software structural model of the radar system. This model was developed as the surveillance portion of an Air Defence System.

In addition to scholarly review on radar simulation systems, nine commercial software tools on radar simulator were examined. The result shows that the most current computer simulators, which are involved in target (aircraft or vessel) detection by radar, might be either a radar simulator or an ATC simulator.

A radar simulator simulates an environment that consists of targets, aircraft or vessels, and a radar system, such as ATC Surveillance Radar System or Marine Radar System. It emulates the functionalities of the radar system as well as provides control over the target's situation, weather condition and terrain. Radar simulators may be used to test the efficiency of radar operation for different radar parameters, operation of systems attached to a radar system such as a RDPS and sometimes the suitability of a radar location. They are also used for training goals. For example, controllers are trained to set the radar parameters for the best operation in different weather conditions.

An ATC simulator simulates most aspects of the ATC's screen and airspace activity. It simulates the environment that a controller deals with, such as radarscope, pilot, weather conditions and even voice communication (Voice Recognition System provides the ability to receive voice commands from the user). ATC simulators are usually used for ATC training; therefore they do not need to simulate every individual part of the ATC system. For example, the system that simulates a radarscope may not simulate the radar system and the RDPS separately. Instead, it may simulate the combination of these two systems. Nevertheless, to achieve more realism in ATC simulators, it is tried to simulate every part of the ATC system.

All of the simulators that are involved in target detection by radar – such as ATC simulators or radar simulators, include two main engines. The first engine simulates targets, i.e., objects that are to be detected by radar. This engine, named Target Simulator Engine, simulates the movement and manoeuvres that targets may take. The second engine simulates the radar operation in detecting targets, named Radar Simulator Engine.

The Target Simulator Engine has almost the same operation in all of these simulators, but different simulators may take different approaches to implement their Radar Simulator Engine. There are two different approaches for the design and implementation of the Radar Simulator Engine:

Approach A: In this approach the Radar Simulator Engine acts as a filter on the output of the Target Simulator Engine. The engine, depending on the target data and radar parameters, reveals some objects as detected targets (if they are in the radar's vicinity) and hides the others as undetected targets. In this approach, if a target is detected by the radar, it will be at the same location as is calculated by the Target Simulator Engine. The radar region is defined by the user and can reflect the effects of terrain and weather conditions.

Approach B: In this approach the Radar Simulator Engine acts as a processor on the output of the Target Simulator Engine. The engine (depending on the target data), radar parameters, and environment conditions (weather and terrain), calculate the azimuth of the target and the slant distance between the target and the radar. This data will be merged with other data from the target, if available, to recalculate the location of the target. In this scenario, the location recalculated by the Radar Simulator Engine may differ from the location calculated by the Target Simulator Engine.

Exercising some current radar simulators reveals that they support training and testing objectives. The simulators for training goals usually employ Approach A, because they do not need to simulate the radar system and the RDPS separately. In contrast, simulators for testing goals usually use Approach B.

Here are some examples of current radar simulators and ATC simulators.

Marine Radar Simulator: This tool is a radar simulator developed by SYDAC Ltd., Australia. It is for training goals for operating marine radar systems [12].

Real Time Radar Simulator (RTRS): RTRS has been developed by Roke Manor Research, UK, to allow the simulation of complex radar and scenarios. Scenario contents and parameters can be varied in real time to allow operators to see the effect of any changes. Detailed terrain, target and radar antenna are used to maximize the realism of the output. This tool is used for both training and testing (radar site evaluation) goals [13].

FIRSTPlusTM: FIRSTPlusTM is developed by Raytheon, Canada. The system consists of two training features - radar simulation and tower simulation. Radar simulation, used primarily in radar training, includes 34 controller and pseudo-pilot positions. Tower simulation includes a three-dimensional tower (five positions, 270 degrees front projection, and three-meter screens), and five two-dimensional tower trainers. This tool is developed for training goals [14].

AeroTrac Simulator: AeroTrack has been developed by Telephonics, USA. It is a flexible tool for training controllers in all phases of air traffic control procedures. It may be also used as an ATC automation system simulator for training technicians and maintenance personnel in support operations. The typical configuration consists of a core simulator server and several pilot position workstations networked on an Ethernet LAN. The AeroTrac SIM outputs radar data in multiple standard formats. This allows the SIM to interface with a variety of operational processing and display systems [15].

ATC Interactive: The simulation program is designed to represent a real ATC environment, and to allow controllers to carry out the control task as realistically as possible. The program uses simple text files to define a simulation scenario [16].

ARTMACS: This simulator is a modular and flexible software that can be used to simulate a wide range of military and civil operational air traffic management systems. It can be used for training on radar and procedural environment. The simulator is used for training purposes [17].

Radar Training Simulator: This radar simulator simulates the operation of radar in target detection. It is designed and developed for marine use. It simulates all functions of small craft radar, models vessels, and wind and rain effects. The simulator is designed for training and provides facilities for trainers to exercise the best radar parameters for the finest target visualization [18].

The RSS simulates the radar system and takes Approach A for the Radar Simulator Engine. In conjunction with the RDPS, the RSS can be used as a radar simulator for both testing and training objectives.

1.6 Contribution of work

Creating a computerized system that simulates the behaviour of ATC surveillance radar is complicated by three matters. The first matter is discovering different natural phenomena that affect the operation of ATC radar system. The second matter is finding some methods and techniques to simulate the effects of these phenomena. And the third factor is considering some techniques to increase the performance of the simulation.

This work addresses these three issues. It explains different effects of some natural phenomena on the operation of ATC radar system. Then it introduces some methods to computerize these effects and some techniques to increase the performance and efficiency of the simulation.

The challenges in the way of developing this simulator and their solutions are encapsulated in a model which indicates the entities of the simulation environment. Using this model not only makes to build a credible simulation system but also adds some

quality attributes such as performance, usability and interoperability to the simulation system.

1.7 Thesis outline

The organization of this thesis is as follows. Chapter 2 describes the ATC radar, their history, types and operation. Chapter 3 covers the simulation environment, including simulation items, their behaviour and interaction, and environmental effects and constraints. Chapter 4 explains the algorithms and techniques that are used in the RSS. Chapter 5 examines the architecture of the RSS. The conclusions are summarized in Chapter 6.

2 Fundamentals of ATC Radar Surveillance

2.1 Introduction

This chapter introduces two different surveillance radar systems that are used in air traffic control. It describes the fundamental principles of ATC surveillance radar along with specific characteristics that may vary between different installations. This information serves as the basis for our design of an ATC radar computer simulation model.

2.2 ATC Surveillance Radar

Two different kinds of surveillance radar are used for air traffic control, Primary Surveillance Radar (PSR) and Secondary Surveillance Radar (SSR).

PSR involves the reflection of a signal transmitted by the radar source. Detection of reflected signals by the radar source is used to generate radar reports. Radar data derived from the PSR is limited to the horizontal direction of the source of the reflection (called the "azimuth") and the distance to the source of the reflection (called the "slant range") based on the elapsed time between transmission of the signal and detection of the reflection.

In the case of SSR, a signal transmitted by the radar source prompts electronic equipment installed on the aircraft to transmit a reply that contains digital information. This information typically includes the aircraft's identification and the aircraft's altitude. Like PSR, SSR also provides information about the slant range and the azimuth of the aircraft.

PSR and SSR can be operated independently. PSR and SSR stations do not need to be synchronized and do not need to be co-located. If they are co-located they are mechanically synchronized. However, this synchronization may not be needed and incoming radar data from different stations are merged together later. Figure 2 [20] is

photograph of a SSR antennae mounted on top of the PSR and Figure 3 [20] is a stand-alone SSR installation.

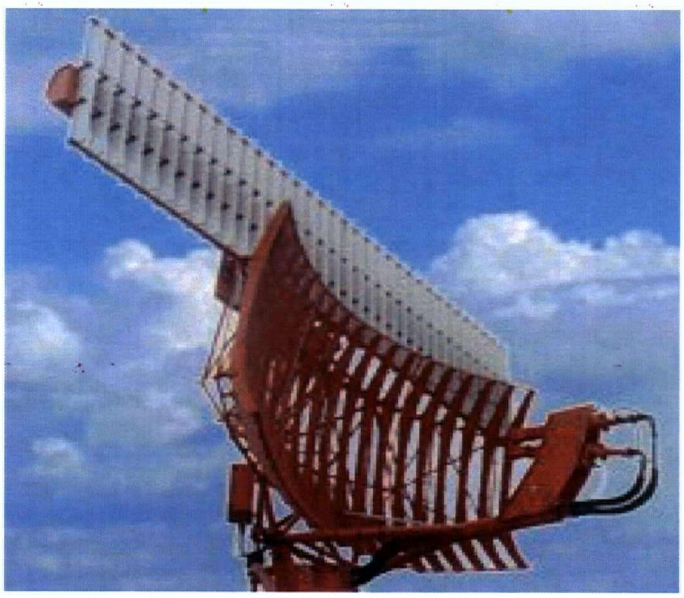

Figure 2: PSR (red antenna) and SSR (white antenna) are co-located.

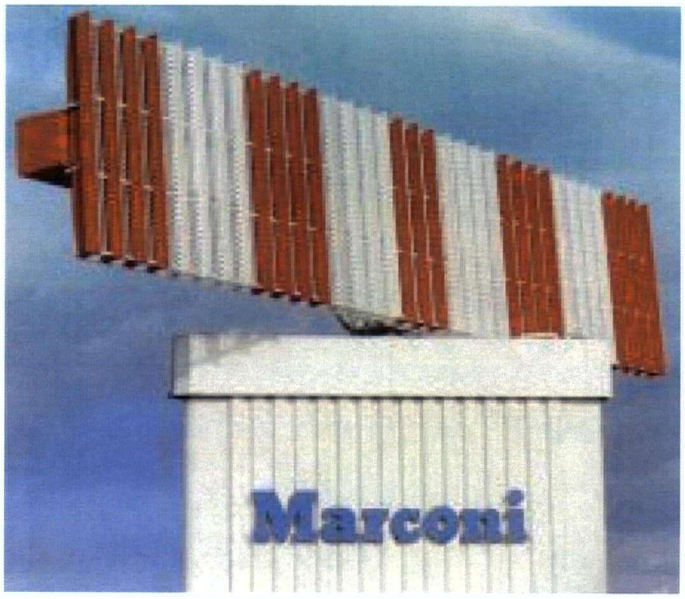

Figure 3: A Standalone SSR System.

Usually the SSR antenna is mounted on top of the PSR antenna. Inevitably there will be some cross coupling of signals between two radars, and careful design is required to ensure the performance of the radars. Setting PSR to operate in **S** band (2700-3400 MHz), using filters, good grounding and careful design on radar installation are some ways to avoid signal interference between two radars.

2.3 Principles of Operation of PSR

In PSR the rotating antenna transmits a high power pulse of energy. When this energy strikes the body of an aircraft, a small amount of it is reflected back towards the antenna. The role of PSR is to detect this small amount of reflected energy. The range of the aircraft is determined by measuring the time space between the transmission of the pulse and the detection of its reflection. PSR also determines the aircraft's bearing by the direction of the antenna.

PSR antennas are made highly directional; that is, more energy is propagated in certain directions than in others. The energy radiated from an antenna forms a field having a specific radiation pattern. A radiation pattern is a plot of the radiated energy from an antenna. This energy is measured at various angles at a constant distance from the antenna.

The shape of the PSR antenna provides specific radiation pattern, so that it can highly focus the propagated energy in one direction, which is called main-lobe or main-beam. However, field leakage makes some low amplitude electromagnetic fields in sides and in back of main-lobe that are called side-lobes and back-lobe respectably. Figure 4 shows the propagation pattern for PSR.

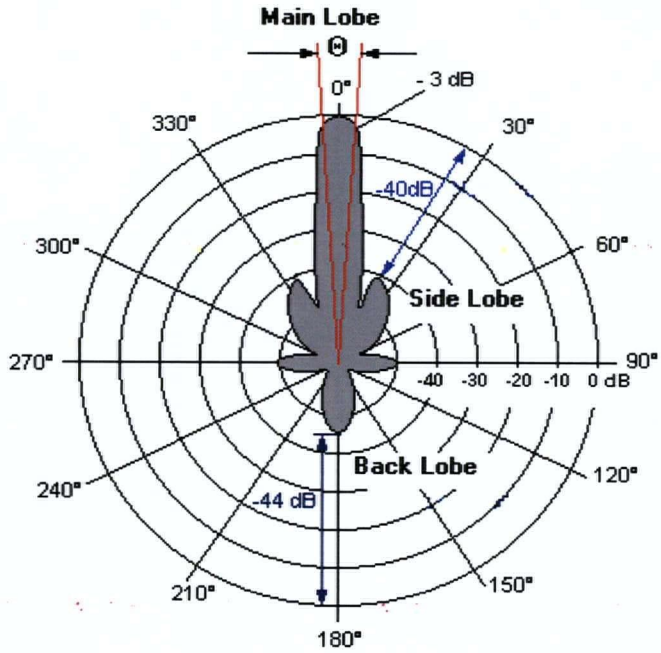

Figure 4: Propagation pattern of PSR Interrogation Signal

PSR has a Timing Schedule for transmitting and receiving signals. The radar transmits a pulse of energy in the transmitting time, which is about 1 microsecond [21]. After the transmission, the radar requires some time to switch from transmitting mode to receiving mode and also to recover its power to send another pulse of energy; this time is called recovery time. After recovery time, the radar waits for signal echoes; this time is called receive time and usually lasts for about 1 millisecond. The receive time has to be sufficiently long to avoid receiving ambiguous signals and echoes from distant aircraft. For example the approximate time for a pulse to be reflected from an aircraft 45 miles away is about 0.5 milliseconds. The receive time interval is followed by a dead time interval as the radar transitions to start of a new cycle. Figure 5 [20] illustrates the PSR Timing Schedule.

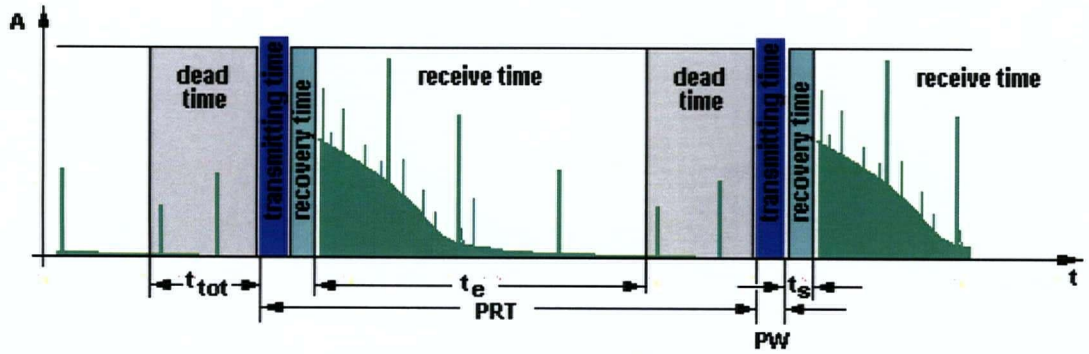

Figure 5: PSR Timing Schedule – (Time Schedule is not uniform.)

The reflection of transmitted pulses are received by the PSR antenna during the receive time and are distinguished as spikes over a signal pattern. The received signal is sent to a Plot Extraction Unit. The Plot Extraction Unit extracts echoes from the signal. Each echo detected by the system at this stage of processing has two attributes: the time elapsed between transmission of the pulse and receipt of its echo, and the horizontal angle of the antenna. The elapsed time is used to estimate the slant range of the aircraft, and the horizontal angle of antenna is used to estimate the azimuth. Figure 4 shows the signal received in receive time by the antenna; the three spikes indicate that three aircraft are detected in the same direction with different ranges.

The probability of target detection for PSR varies with weather condition, rotation rate and frequency of pulse transmission. Typically, the probability detection for the PSR is above 99% [21].

The accuracy of PSR in range and azimuth measurement depends on the radar specification. Typically, the accuracy of a PSR in range measurement is about 150 meters for a target at the distance of 90 kilometres. The accuracy of the PSR in azimuth measurement is 0.035 degree [22].

The advantage of Primary Surveillance Radar (PSR) is that it operates totally independently of the aircraft - that is, no action from the aircraft is required for it to provide a radar return.

The disadvantages of PSR are that, firstly, enormous amounts of power must be radiated to ensure returns from the target. This is especially true if long range is desired. Secondly, because of the small amount of energy returned to the receiver, returns may be easily disrupted due to such factors as changes in target attitude or signal attenuation due to heavy rain. This may cause the displayed target to fade. Thirdly, the correlation of a particular radar return with a particular aircraft requires an identification process. When PSR was the only type of radar available, this was typically achieved by the Controller observing a directed turn by the aircraft, or by correlating DME's (Distance Measurement Equipment) distance report by the aircraft with the position of a particular return along a known track [19]. Finally, PSR does not provide any information about the altitude. Without altitude, there is not enough data to precisely determine the location of the target. (Slant range, altitude and ground range form three sides of a right triangle. To determine the ground range precisely, we need both slant range and altitude.) However, there are some ways to estimate the location of the target by approximating its altitude, but it is highly preferable that PSR replies are integrated with SSR replies [19].

2.4 Principles of Operation of SSR

The disadvantages of PSR outlined above led to the employment of another aspect of radar development. This involved the Identification Friend or Foe (IFF) system and a means to send data about the aircraft's altitude. The system, which became known as Secondary Surveillance Radar (SSR), or in the USA as the Air Traffic Control Radar Beacon System, relies on a piece of equipment aboard the aircraft known as a "transponder."

The signal transmitted by the SSR ground station is called an interrogation signal. The SSR station can request different kinds of data from an aircraft's transponder using different interrogation modes. The interrogation mode is signaled by means of three pulses, P1, P2 and P3. In particular, the interrogation mode is indicated by the elapsed

time between the P1 pulse and the P3 pulse. For example, if the timing space between P1 and P3 is 8 microseconds the interrogation mode is Mode A Identification. This particular interrogation mode is used to request the aircraft's identification code. If the elapsed time between P1 and P3 is 21 microseconds, the interrogation mode is Mode C Pressure-Altitude. Figure 6 [23] illustrates the interval between P1 pulse and P3 pulse in SSR interrogations.

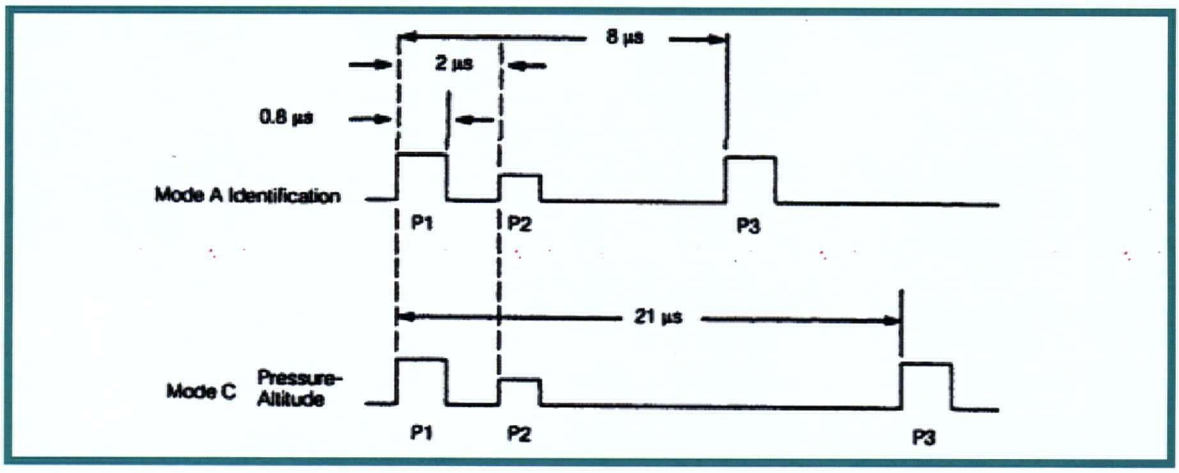

Figure 6: The interrogation Mode is determined by time space between P1 and P3.

Regardless of the interrogation mode, all three pulses are 0.8 microseconds wide. The frequency for all interrogations is 1030 MHz.

The P2 pulse is used by the transponder to determine whether the interrogation is received from the main beam or a side-lobe of SSR's radiation pattern. A reply to a side-lobe interrogation gives the controller an erroneous indication of the aircraft's position. For this reason, side-lobe suppression (SLS) is used to inhibit the transponder's reply in response to a side-lobe interrogation [19].

The three-pulse SLS interrogation method uses a directional radar antenna that transmits a pair of pulses referred to as P1 and P3 pulses. Two microseconds after the P1 pulse is transmitted from the directorial antenna the second pulse, P2, is transmitted from an omni-directional antenna. (Omni-directional antenna propagates signals with the same power in all directions.) The P2 pulse is used as a reference pulse for SLS determination.

The signal strength of the omni-directional P2 pulse is just sufficient to provide coverage over the area that side-lobe propagation presents a problem. Figure 7 [23] shows the propagation pattern of the SSR to transmit P1, P2 and P3 pulses [23].

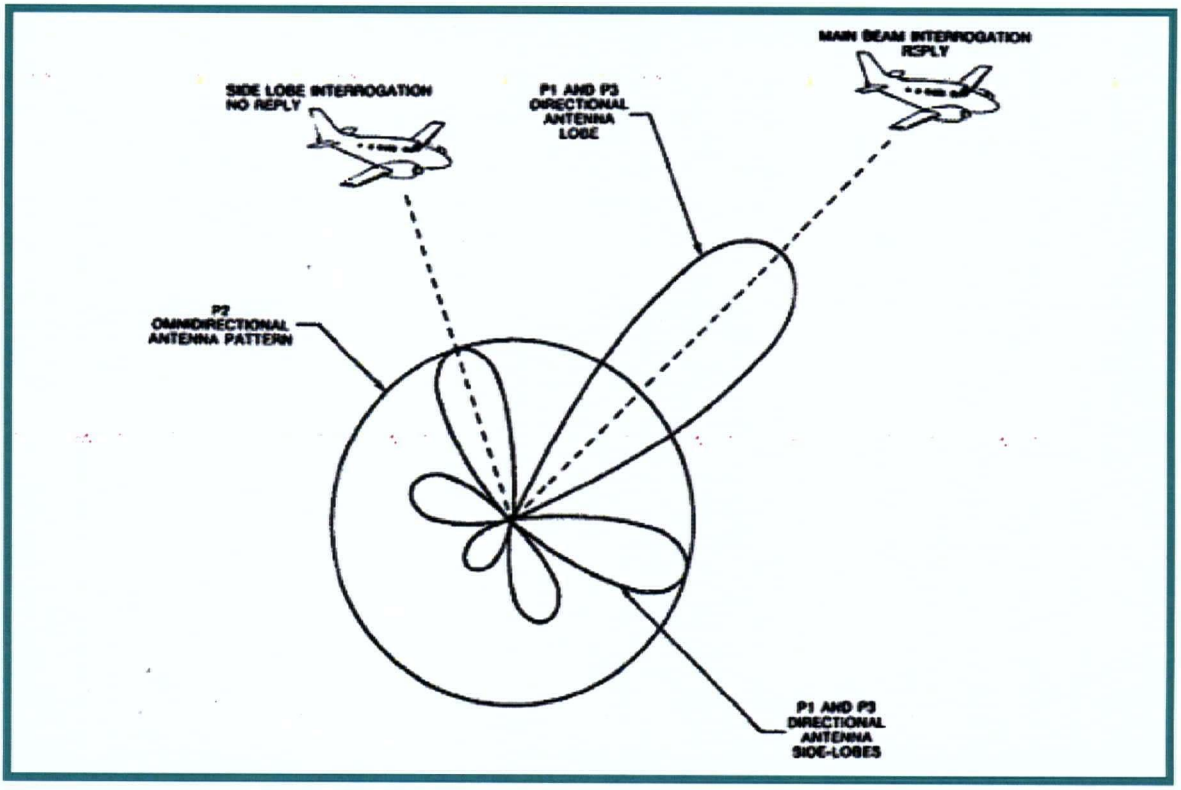

Figure 7: Propagation pattern of SSR Interrogation Signal

The airborne transponder SLS circuit detects side-lobe interrogation by comparing the amplitude of the P2 pulse in relation to the P1 pulse. When the omni- directional P2 pulse is equal to or greater than the directional P1 pulse, no reply will be generated. Identification of the side-lobe interrogation is established before the P3 pulse is received. A valid main-lobe interrogation is recognized when the P1 pulse is at least 9dB larger than the P2 pulse [23]. Figure 8 shows the conditions that the transponder replies to the interrogation. If powers of all pulses are the same, the interrogation is rejected but if the power of P2 is 9db less than that of P1 and P3 the interrogation is accepted.

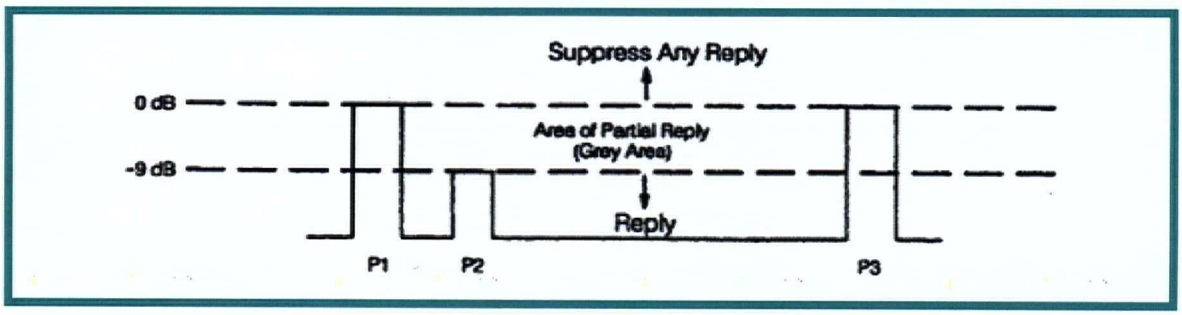

Figure 8: Side-Lobe Detection and Reply Suppression

Although transponders are able to decide whether an interrogation is coming from the side-lobes or the main-lobe of the SSR antenna, but SSR is not able to distinguish between replies received through the main-lobe and replies received through the side-lobes. SSR receives signals from its side-lobes. These signals may be the result of signal reflection or may be transmitted by other transponders. Received signals from the side-lobe of the SSR antenna results in extraneous replies [19].

Reply signals are generated by the transponder when an interrogation signal is determined to be valid. A transponder delays its reply by $3 \pm 0.5 \mu s$ timed from the interrogation pulse p3 [19].

The coded reply signal is composed of a series of pulses transmitted on a carrier of 1,090 plus or minus 3 MHz. It consists of various arrangements of code pulses within the boundaries formed by the two framing pulses, F1 and F2 (Figure 9). Regardless of the mode of operation, these framing pulses are always present in the coded reply signal and are spaced 20.3 microseconds apart.

The reply code is divided into four pulse groups labelled A, B, C, and D. Each group contains three pulses with suffix 1, 2 and 4 to give the total number of 12. The SSR reply indicates an octal number ABCD, which A is the sum of suffixes of group A pulses, as well as others. For example if the reply code consists of A1 and A4 pulses, A would be 5. The assigned reply code-0000 would cause no pulses to appear between the framing pulses, and code 7777 would result in all 12 pulses to be present between FI and F2.

Figure 9 shows the reply format of an interrogation. This format is similar to replies of all interrogation modes.

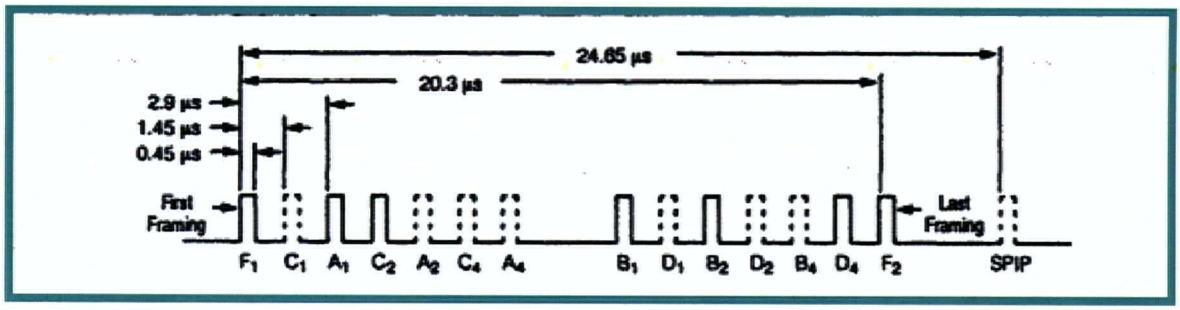

Figure 9: Reply signal format

The Special Position Identification Pulse (SPIP), initiated upon request of the controller, is generated by momentarily depressing the IDENT button located on the transponder control head. The SPIP causes a special effect on the controller's screen that aids in determining the aircraft's position. This pulse occurs 4.35 microseconds after the last framing pulse (F2) and is transmitted with each Mode A reply for 15 to 20 seconds after releasing the IDENT button.

The probability of target detection for SSR varies with weather condition, rotation rate probability of transponder reply (which is about 90%) and the beam width. SSR is designed to have the probability detection of 99% [19].

The accuracy of the SSR in range and azimuth measurement depends on the radar specification. Typically, the range measurement in the SSR has about 100 meters range bias and 1m/NM range gain. The azimuth measurement in the SSR has 0.1 degree azimuth bias [23].

2.5 Plot Extraction Unit

Radar produce multiple replies for one aircraft in a single rotation. (A reply refers to a transponder response, in the case of the SSR, or a reflection in the case of the PSR that is received by radar.) In general, for avoiding high false detection rates, the number of replies per target must be kept high.

PSR and SSR replies are sent to different Plot Extraction Units. The Plot Extraction Units produce Radar Reports from these replies and send them to the Plot Combiner for PSR and SSR data combination. Figure 10 shows the radar system from the radar antenna to the Radar Data Processing System (RDPS).

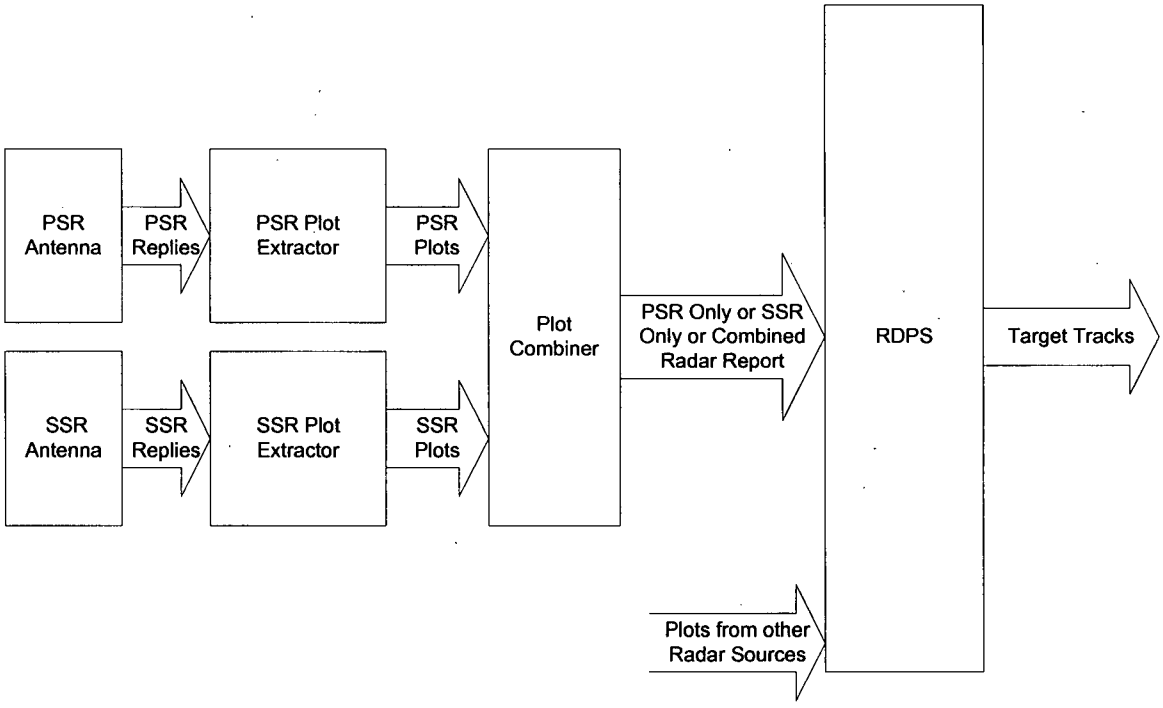

Figure 10: The Radar System and the RDPS

The plot extraction is performed using two different methods, Sliding Window and Monopulse. These methods are used for Secondary Surveillance Radar. For Primary

Surveillance Radar, the Plot Extraction Unit is as the same as the Sliding Window in the SSR with less complexity and fewer functions.

2.5.1 Sliding Window

The Sliding Window Plot Extraction method is used for standard SSR and includes the following procedures: [19]

1. Reply detection: This function is to detect individual reply pulses and recognize the whole reply. It performs pulse detection, bracket detection and code extraction. It identifies garbled, phantom (false), military and emergency replies. It also performs mode determination and range measurement. The mode determination is used for identifying a reply as a response to the transmitted interrogation.

2. Defruiting: All the received replies are not synchronized with the transmitted interrogation; a reply is synchronized with an interrogation when it is the response to it. A synchronized reply is received through the main-lobe where the interrogation has been transmitted. The additional replies, which are caused by multipaths, other transponders or received via the side-lobes, are called fruits. This function eliminates fruits from radar replies.

3. Target detection and position measurement: This part is to calculate all necessary information from radar replies to produce a Radar Report. The Radar Report includes the azimuth and the altitude of an aircraft.

4. Reply correlation: After the false replies have been filtered out, the range and the azimuth of the remaining replies (for replies to the same interrogation mode) are compared to locate groups of replies which have originated from the same aircraft. Two replies are correlated if they lie in one range-azimuth zone.

5. Combining SSR reports with PSR reports: The limitations of SSR are different than the limitations of PSR, and therefore it is beneficial to combine these two sources of surveillance information. This combination is usually applied if two reports are placed in the same range-azimuth zone. If the reports are not correlated, both PSR and SSR reports will remain unchanged. The reports then are sent to the Radar Data Processing System (RDPS) for track update or initiating a tentative target.

2.5.2 Monopulse

The Monopulse Plot Extraction Unit performs all the functions of Sliding Window Plot Extraction Unit but with considerably improved precision. In this technique the radar takes advantage of direction-of-arrival measurement for every receiving pulse. This information is very useful in determining the bearing of aircraft and they provide critical assistance in decoding SSR replies, especially in the case of overlapping replies.

In nonmonopulse SSR, the plot validation is achieved by repetition, i.e., by requiring a number of replies from each aircraft for each interrogation mode. When overlapping replies are detected, the whole reply is labelled as suspect and not used. In contrast, in a monopulse plot extractor, a garble bit is assigned to each garbled pulse and this allows using ungarbled pulses. This technique causes monopulse plot extractors to need fewer radar replies than nonmonopulse plot extractors to produce reliable radar returns.

The Radar Report produced by the Plot Extraction Unit consists of following data fields [19]:

1. Azimuth: this field contains the direction of coming reply.
2. Identity 1: this field holds the aircraft's identification.
3. Height: this field holds the height of aircraft's obtained from Mode C interrogation.
4. Emergency: this field is set when the SSR code 7700 is transponded.

5. Communication failure: this field is set when the SSR code 7600 is transponded.
6. Hijack: this field is set when the SSR code 7500 is transponded.
7. Military emergency: this field is set when a military emergency reply has been received.
8. SPI: this field is set when replies contain SPI for the last pulse in every reply.

2.6 ATC Surveillance Radar Characteristics

There are some radar antenna characteristics that define the performance of radar in target detection.

Elevation Angle: The elevation angle is the angle between the horizontal plane and the line of sight, measured in the vertical plane.

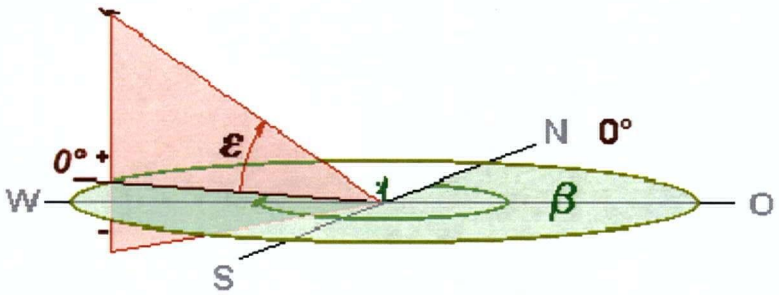

Figure 11: Elevation angles

The elevation angle of surveillance radar depends on the antenna size and frequency of transmitter. The maximum elevation angle for SSR is typically 12.5 [19] degrees; usually below 20 degrees [19]. The maximum elevation angle for PSR depends on the number of feed horns. A feed horn is a device to guide signals into radar antenna and is located in front of it. Figure 12 [23] shows two feed horns of a PSR antenna.

Figure 12: Two feed horns are in front of the PSR antenna

The more feed horns the radar has the more elevation angle it can reach. Most PSR antennas have two feed horns with a maximum elevation angle of about 20 degrees. Some complex antennas with more feed horns may reach a maximum elevation angle of 40 degrees.

The fact that the maximum elevation angle of radar antenna is less than 90 degrees causes the existence of an area called "a cone of silence." The cone of silence is a circular region above the radar antenna in which aircraft are not seen by radar. The area of a cone of silence depends on the elevation angle and the altitude of the aircraft. For example the cone of silence for an aircraft, which is flying at 10,000 feet above a radar antenna with maximum elevation angle of 40 degrees, is a circle with the radius of 2.26 miles.

Slant Range: Slant range is the distance that a beam of energy travels between radar antenna and the target. The maximum slant range of PSR and SSR differs for different products. The typical value of a maximum slant range for both PSR and SSR varies between 60 and 250 nm (Nautical Mile), is usually 250 nm for SSR and 90 nm for PSR [19].

SSR usually has a greater slant range than PSR because transmitted signals by SSR suffer from one-way power loss while transmitted signals by PSR suffers from two-way power loss.

The maximum slant range of SSR depends on the power of its transmitter. The maximum slant range of PSR depends on both transmitter power and receive-time.

The targets must be at a minimum range from the PSR antenna to be detected; because it is necessary that the sending impulse leaves the antenna completely and the radar unit switches to receiving mode. This minimum measuring range (R_{min}) is the function of pulse width and recovery time and specifies the minimum slant range for the PSR [20].

$$R_{\min} = \frac{PW + recoverytime}{12.35\mu s}$$

For example for PW = 1 μs and recovery time = 1 μs, the $R_{\min}$ would be 0.16 nm.

It is reasonable to expect that the SSR operation is similarly bounded by a minimum slant range. However, we were unable to find any published information that confirms this. Nevertheless, since SSR and PSR are generally synchronized with each other to obtain coherent information from a target, here it is presumed that the minimum slant range of SSR is the same as the minimum slant range of PSR.

Rotation Rate: The rotation rate of PSR and SSR varies for different products and usages. The typical value of rotation rate or scanning frequency is between 4 to 15 rpm [19]. Long distance radar usually operates with low rotation rates.

Beam Position: A Beam Position is one of the fixed positions in the rotation of the radar on a horizontal plane at which the radar may receive radar replies. Existing ATC surveillance radar usually have 4096 beam positions. Since the beam width of radar is

about 1 degree and the number of interrogations in each beam width is between 10 and 20 interrogations, so it requires that the number of beam positions is about 3600 to 7200. Typically the number of beam positions varies between 900 and 7200 for different beam widths and accuracy level for azimuth and range measurements [19].

Beam Width: Beam Width is the angle on a horizontal plane subtended by the beam of a radar antenna at one beam position. The beam width of PSR and SSR has a value between 1 and 3 degrees.

The beam width is associated with the main-lobe width. It is the angle between the extreme right and the extreme left beams. The beam width of radar is larger for close targets than for targets at a greater distance. The reason for this is that close aircraft may be detected by the side-lobes as well, so the beam width contains not only the main-lobe but also the side-lobes.

2.7 Conclusion

This chapter covers the details of radar systems from radar antenna to the Plot Extraction Unit, which are important in modelling radar systems. It described the operation of two ATC surveillance radar systems, PSR and SSR. It explained some radar antenna characteristics such as elevation angle, beam width and radar rotation. It also gave details about the Plot Extraction Unit and its role in a radar system. This detail provides a foundation for the development of the simulation model described in the next chapter.

3 A Simulation Model for Radar Surveillance

3.1 Introduction

This chapter describes a simulation model for the Radar Source Simulator (RSS). Some elements of this model, such as aircraft representation, can be anticipated without an in-depth understanding of ATC operation and technology. However, other elements of this model are considerably less obvious without this in-depth understanding. For example, our model takes into account the environment effects such as signal refraction. As described in the previous chapter, this research is based on a thorough investigation of the ATC's operation and technology. Based on the results of this investigation, the model presented in this chapter has been developed and refined. Replication of all the natural phenomena involved in the transmission and reception of radar signals would far exceed the capacity of any computer-based simulation. Instead, we have identified particular aspects of these natural phenomena that have the potential to significantly affect the overall accuracy of ATC radar data, and we have proposed behaviour abstractions that approximate the effects of these natural phenomena.

3.2 Actor Model

Before presenting a model for our system, it is first important to introduce a metamodel called actor model [24].

The actor model has a component-based approach for modeling a system. The component-based approach provides a manageable model to support heterogeneity in modeling complex system. For example, in an electromechanical system there are some mechanical parts that follow Newton's laws and there are some electrical parts that follow Krichoff's laws. To model this system, it is better to separate mechanical parts from electrical parts and then model each part as a separate component. This method of modeling removes the complexity of interaction between various electrical and

mechanical parts. It also enables engineers to focus on their engineering domains and ignore information of less interest.

In addition to having a component-based approach in modeling, the actor model has another characteristic specification that distinguishes it from other models. This model is not a unified model that supports all interactions between its components. Instead of mapping all components into a grand unified model, the actor model uses existing models. In other words, in the actor model we create components from atomic components, which present primitive models and compose smaller components to create more complex ones. This approach helps to significantly decrease the diversity of different styles in component integration, because the integration styles are defined at very low levels between atomic components.

The actor metamodel is composed of three basic building blocks.

1. Actors
2. Ports
3. Directors

Actors encapsulate executions and provide communication interfaces. Actors can be active or passive. Active actors can change the states of other actors or the states of themselves. Active actors can also initiate events and trigger other actors or themselves. Passive actors do not change the state of other actors nor trigger other objects. They only can process events and change their own state.

Ports represent the communication between actors. They are the actors' interfaces. Ports exchange data by asynchronous message passing; meaning that neither the receiver nor the sender is blocked during the communication. Actors can have as many ports as is necessary.

Directors define the semantics of messages passing between ports. They implement the integration styles between actors.

The actor model has the following notations.

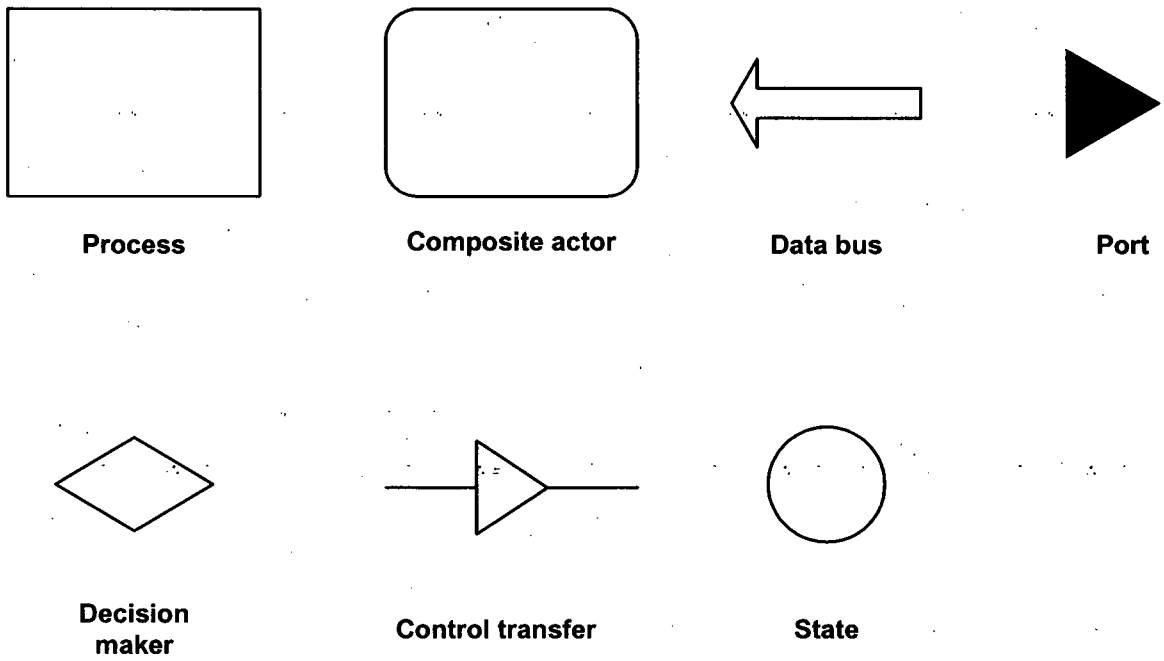

Figure 13: Notations used in the actor model

In an actor model graph, a process represents a set of actions that are applied to data. A state represents a status of a variable. A control transfer controls the flow of data from one point to another and a decision maker decides the direction of the data flow.

3.3 RSS Modelling

The RSS model includes five actors. Each of these actors models a real entity from RSS environment. These entities are:

1. Radar Transmitter
2. Radar Receiver
3. Obstruction
4. Aircraft
5. Environment

The actor model that represents these entities and interaction between them is shown in Figure 14.

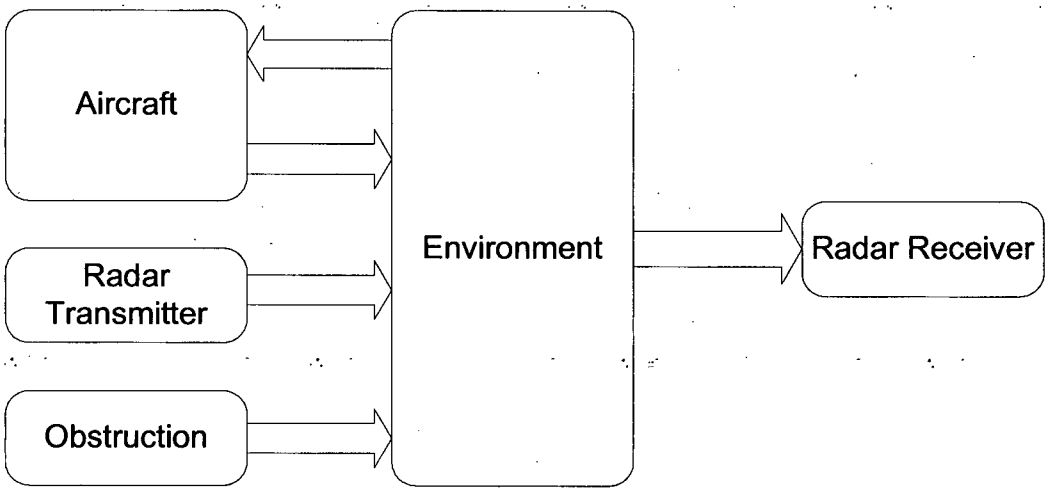

Figure 14: The RSS model

The Aircraft Actor has the following main responsibilities:

1. To determine the trajectory (position, altitude, speed, bearing at any point in time) for a particular aircraft based on a scripted sequence of actions (e.g., changes in altitude).

2. To determine the status of the transponder at any point in time (i.e., off, busy or ready to respond).

3. To determine the Mode C altitude that would be reported by the transponder in response to the interrogation Mode C altitude (as a function of ambient barometric pressure which, in turn, is a function of the true altitude and a set of altimeter settings).

4. To maintain a list of station observations of barometric pressure.

Responsibility #4 may be unintuitive because the geographic locations associated with the observations are physically outside the aircraft. But since the altimeter settings depend on these observations, the aircraft must be aware of the observations from stations to calculate the altimeter settings.

The Aircraft Actor initiates events to update its states. It also receives triggers from the Environment Actor. The Aircraft Actor is an active actor.

The Radar Transmitter Actor has the following main responsibilities:

1. To determine the operational status of the radar transmitter (i.e., on/off).
2. To determine the stream of PSR pulses, including a characterization of the range, direction, position and maximum elevation angle of each pulse.
3. To determine the stream of SSR interrogation signals including characterization of the kind (mode A and Mode C altitude), range, direction, position and maximum elevation angle of each signal.

The Radar Transmitter Actor is an active actor. The actor initiates events to change its states. It also triggers the Environment Actor.

The Radar Receiver Actor has two main responsibilities:

1. To correlate and merge radar replies to generate PSR, SSR and combined PSR/SSR radar returns.
2. To consider the target detection probability and transponder reply probability in producing radar returns.

The Radar Receiver Actor is a passive actor. The actor does not have any internal state. The actor only processes the data that it receives and sends it to the output.

In the physical world, radar signals travel between the radar and the aircraft through a physical environment. This environment is not a passive medium. Instead, the environment actively affects the transmission of radar signals. For example, the environment imposes refraction on signals or presents the multipath between the radar and the aircraft. To take into account the role of the environment, we have a distinct actor for the environment, the Environment Actor. The Environment Actor is the sole interface between the Aircraft and the Radar Transmitter Actor. The Environment Actor has the following main responsibilities:

1. To determine which, if any, aircraft are illuminated by a given PSR pulse transmitted by a radar transmitter and, if so, to determine the slant range.

2. To determine which, if any, aircraft respond to a given SSR interrogation signal, and, if so, to determine the slant range.

The Environment Actor is a reactive actor. The actor only reacts to the events received from other actors. The internal states of the actor change over receiving events from other actors.

The Environment Actor must also take into account:

1. The effects of the signal refraction on whether or not a signal illuminates the aircraft as well as its effect on determining slant range.

2. The effects of the earth's curvature on whether or not a signal illuminates the aircraft as well as its effect on determining slant range.

3. The possible obstruction of a signal by the terrain such as a mountain or tall building.

Obstructions are presented by the Obstruction Actor. The Obstruction Actor is a passive actor and its internal status does not change. This actor provides detailed information about obstructions to the Environment Actor. The Obstruction Actor's main responsibility is:

1. To provide static details about obstructions.

In this model, Aircraft Actor, Radar Receiver Actor, Radar Transmitter and Obstruction Actor are independent actors and there is no direct interaction between them. It means that they can be changed without any side effect on other actors. The interaction between these actors occurs through the Environment Actor. The outcome of the Environment Actor indicates if an aircraft and radar are connected to each other, which results in detection of aircraft by radar. All the rules that relate an aircraft to radar are encapsulated in the Environment Actor.

3.4 Aircraft Modelling

RSS simulates the trajectory of aircraft by approximating the realistic dynamic motion of an aircraft. The simulation supports possible manoeuvres of the aircraft in terms of bearing, altitude and speed changes. It also simulates the action of an onboard transponder by responding to ground radar interrogations.

An aircraft is modeled as a continuous-time component, as its state variables changes continuously with time. These state variables define some aircraft dynamic characteristics, which are used to implement an aircraft's system.

The aircraft model has the following state variables:

1. Aircraft position: the position of the aircraft with its latitude and longitude.

2. Aircraft bearing: the angle, in a clockwise direction, on the horizontal plane between true north and the direction of the radar beam.

3. Aircraft bearing rate change: the absolute value of the rate at which the bearing of the aircraft increases or decreases in response to change bearing.

4. Aircraft ground speed: the rate at which the ground distance is traveled over time.

5. Aircraft acceleration: the absolute value of the rate at which the ground speed of the aircraft increases or decreases in response to a change in ground speed.

6. Aircraft altitude: the vertical distance between the ground and the aircraft.

7. Aircraft altitude change rate: the absolute value of the rate at which the altitude of the aircraft increases or decreases in response to a change in altitude.

8. Aircraft transponder status: whether the transponder of the airplane is on or off. If the transponder is off, it does not reply to the SSR's interrogations.

9. Aircraft identity: the identity of the airplane. This identity is sent to the SSR in response to Mode A identification code that is one of the SSR's interrogations.

10. Aircraft Mode C altitude status: whether the transponder replies to the Mode C interrogation or not.

11. Aircraft Mode C altitude: it determines the transponder's answer to the interrogation Mode C. The SSR sends interrogation Mode C to inquire about the aircraft's altitude. By receiving this interrogation, the transponder sends back the Mode C altitude. The Mode C altitude is obtained from airplane altimeter.

3.4.1 Aircraft Actor

The Aircraft Actor has two input ports, P1 and P6. One of them receives discrete events, which are created by the user, and the other one receives triggers from the Environment Actor, which initiates executions to generate the output of the Aircraft Actor. The discrete events that are received by the Aircraft Actor are:

1. Aircraft bearing modification (E0)
2. Aircraft ground speed modification (E1)
3. Aircraft altitude modification (E2)
4. Aircraft transponder status modification (E3)
5. Aircraft Mode A SSR code modification (E4)
6. Aircraft Mode C Altitude modification (E5)

The Aircraft Actor also controls four output ports. The output of the Aircraft Actor goes to the Environment Actor. These ports are:

1. Aircraft position (longitude and latitude). (P2-P3)

2. Transponder settings indicate whether the aircraft transponder is 'on' or 'off,' if it is 'on' then it replies to the radar interrogation Mode C Altitude (Mode C Status variable). (P4-P5)

Figure 15 shows the Aircraft Actor's first level of the hierarchy.

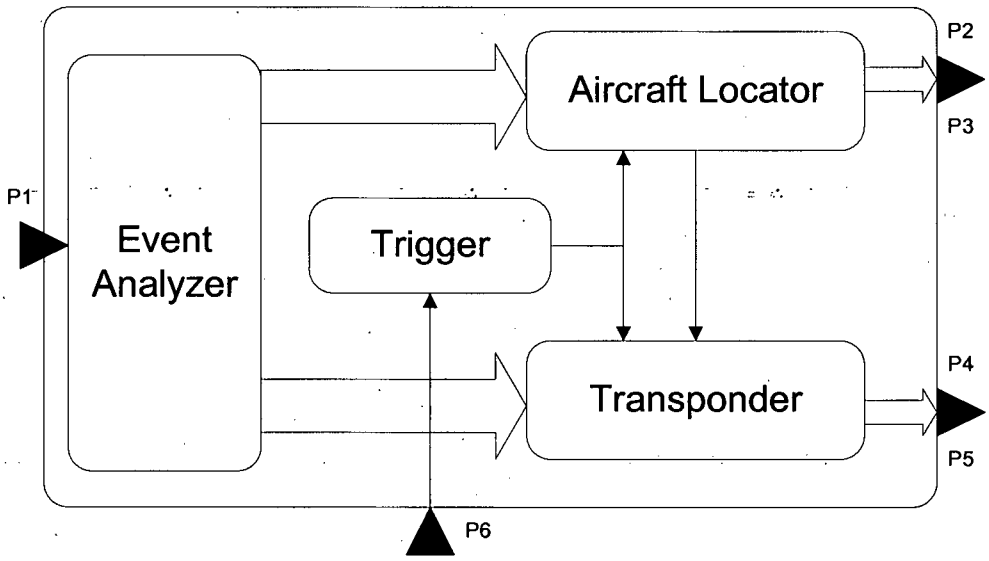

Figure 15: Aircraft Actor

The Aircraft Actor has a three level hierarchy. On the first level, the Aircraft Actor consists of four sub-actors: Event Analyzer, Trigger, Aircraft Locator and Transponder.

3.4.2 Event Analyzer

This actor implements the event messages that are sent to the Aircraft Actor. The received events are separated based on their messages and dispatched to the appropriate actors. Figure 16 shows the Event Analyzer Actor.

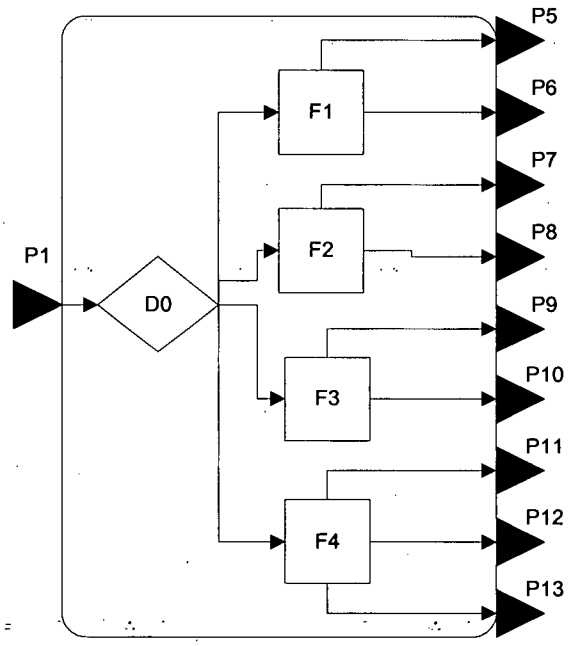

Figure 16: Event Analyzer Actor

The Event Analyzer Actor consists of one decision-maker D0 and four functions F1 – F4. These functions perform the following actions (Events E0-E5 are described in section Aircraft Actor):

1. D0 decides if an event is E0, E1, E2 or E3-E5.
2. F1 extracts the final bearing and the bearing change rate from E0.
3. F2 extracts the final ground speed and the ground speed change rate from E1.
4. F3 extracts the final altitude and the altitude change rate from E2.
5. F4 separates events E3, E4 and E5 from each other.

The result will appear on output ports P5-P13. Ports P5-P10 are connected to the Aircraft Locator Actor to provide information about speed, altitude, acceleration and other aircraft's motion characteristics. Ports P11-P13 are connected to the Transponder Actor to provide transponder settings.

3.4.3 Aircraft Locator

This actor implements the commands to locate the aircraft on the globe. It calculates the latitude and the longitude of the aircraft based on its groundspeed and bearing. It also computes the aircraft's current altitude, groundspeed and bearing based on their initiate values and changes during the simulation. The composite Aircraft Locator Actor consists of five atomic actors. Figure 17 shows the Aircraft Locator Actor and its components.

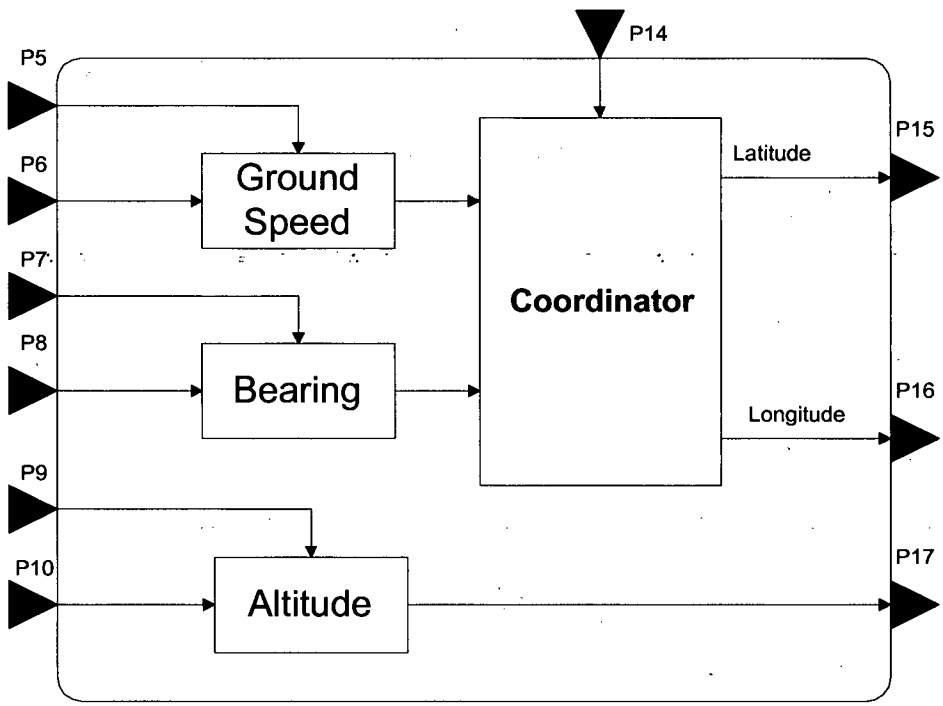

Figure 17: Aircraft Locator Actor

The actor's Groundspeed, Bearing and Altitude maintain the current dynamics of the aircraft. Each of them has two inputs related to the final value of the subject and the subject's rate of change. The output of the actor is the current value of the subject. For example, the Groundspeed Actor receives two inputs from its ports P6 and P7, the final groundspeed of 500 nm and the acceleration of 5 nm/m/h. The output of the Groundspeed Actor is the current groundspeed of the aircraft, which increases from its initial value to 500 nm with the corresponding acceleration. The Groundspeed, Bearing and Altitude actors follow the following rule:

$$current - value(t + h) = change - rate \times h + previous\_value(t)$$

Current groundspeed and bearing are used to project the aircraft on the surface of the earth in the Coordinator Actor.

3.4.4 Coordinator Actor

The Coordinator Actor calculates the aircraft's position. It contains functions to produce the trajectory of an aircraft by having a given starting point and motion characteristics. It calculates the distance traveled and the aircraft's heading direction to determine the trajectory. This actor has an input port P14 to receive triggers. When the actor is triggered, it calculates the next position of the aircraft in the latitude/longitude grid and sends them to the output. The output of the Coordinator Actor is directed to the Aircraft Locator output ports.

3.4.5 Positioning

The position of an aircraft on the earth is identified by its latitude and longitude. The concept of positioning is to determine the new position of an aircraft with its initial position, travelled distance and the direction of traveling. For example, if the position of an aircraft is 30 degrees latitude and 40 degrees longitude and it travels 200 kilometres with the azimuth of 45 degrees, the new position of the aircraft would be 31.24 degrees latitude and 41.47 degrees longitude.

The method for determining position depends on two basic parameters, namely, the shape and dimensions of the earth. In particular there are two models of earth, ellipsoid model and spherical model. The ellipsoid model is more accurate than the sphere model because it takes account the unevenness in the radius of the earth. But the sphere model is computationally simpler than ellipsoid model. Fortunately, the relative inaccuracy of the sphere model is not significant for the purpose of this simulation model. The maximum error between using the sphere model (with radius of 6371 km) and the ellipsoid

(WGS84) model for positioning purposes is about 100 meters for every 100 km. Therefore, because the error between the sphere and ellipsoid model is minimal and the fact that using a sphere model is more computationally efficient and simpler, we choose the sphere model of the earth for positioning purposes. Therefore, we assume that the earth is a perfect sphere and its radius of the earth is constant.

In order to perform positioning, the Coordinator Actor calculates the distance that is traveled by the aircraft using the following equations:

$$dis(t+h) = \tfrac{1}{2} \times acceleration \times h^2 + ground\_speed(t) \times h$$

It then computes the new latitude (Lat2) and longitude (Lng2) using the following equations:

$$GCA = dis/\mathrm{Re};$$
$$\sin lat2 = \cos(bearing) \times \sin(GCA) \times \cos(Lat1) + \cos(GCA) \times \sin(Lat1);$$
$$Lat2 = a\sin(\sin Lat2);$$
$$\cos dL = \frac{\cos(GCA) - \sin(Lat1) \times \sin(Lat2)}{\cos(Lat1) \times \cos(Lat2)};$$
$$Lng2 = Lng1 + sign(180 - bearing) \times a\cos(\cos dL);$$

The model includes extra calculations to adjust the latitude and longitude when the aircraft is passing the poles.

After finding Lat2:

```
if ( Lat > 89.90 )
      {
         if ((bearing < 90) || (bearing > 270) )
            {
               bearing = bearing - 180;
               bearing = adopt_bearing(bearing);
```

```
                if (bearing_change_rate > 0)
                   final_bearing = adopt_bearing(final_bearing +180);
                else
                    final_bearing = adopt_bearing(final_bearing -180);

            //pole is passed
            pole_passed = 1;
               }
        }

if ( lat < -89.90 )
        {
            if ((bearing > 90) && (bearing < 270) )
            {
                bearing = bearing - 180;
                bearing = adopt_bearing(bearing);
                if (bearing_change_rate > 0)
                   final_bearing = adopt_bearing(final_bearing +180);
               else
                   final_bearing = adopt_bearing(final_bearing -180);

            //pole is passed
            pole_passed = 1;
               }
        }
```

After finding Lng2:

```
if (pole_passed == 1) lng -= 180;

if (lng > 180.0) lng -= 360;

if (lng <= -179.999) lng += 360;
```

3.4.6 Transponder Actor

The Transponder Actor is a composite actor in the second level of the Aircraft Actor's hierarchy. This actor models the aircraft's transponder in response to SSR interrogation Mode C Altitude. It also reveals the identity of the aircraft in response to the interrogation Mode A.

The altitude information transmitted by a transponder in response to a Mode C interrogation signal is a measurement of the air pressure at the aircraft's altitude. This measurement is encoded as an altitude value. Since the air pressure at a constant elevation above sea level can vary over time and position, the simulation of the ATC radar data with Mode C altitude information must take into account these variations to provide a realistic simulation of ATC radar. These variations can be significant. The Mode C Altitude can vary in response to changes in air pressure by several thousands of feet for an aircraft flying at a constant height above sea level. For example, an aircraft flying at 10,000 feet above sea level may realistically report a Mode C Altitude anywhere in the range of 6,000 feet to 12,000 feet above sea level. The ground-based RDPS uses a correction factor based on barometric pressure at ground level to convert the Mode C Altitude into a value that more accurately indicates the height of the aircraft above sea level.

In principle, the simulation tool could include a detailed atmospheric model that simulates variations in air pressure. However, such an approach is likely to require substantial computational power. Instead, we have used a simpler approach of allowing the user to provide the simulation tool with time-based observations of station pressure. Each observation includes the observed station pressure, the position of the observation (in latitude and longitude) and the height of the observation (in feet above sea level). With a minimum of three such observations, the pressure altitude encoded as Mode C Altitude can be determined as follows:

1. Convert each observation of the station pressure to an altimeter setting. With the new altimeter setting, the aircraft will indicate the altitude of the station above sea

level, assuming the aircraft is on the ground at the same location the pressure value was initially determined.

2. If less than three station pressure observations are available, then use a standard pressure setting, i.e., 29.92 inches Hg, as the derived altimeter setting in Step 3. Otherwise, use the altimeter settings for the three nearest station pressure observations to derive, by means of interpolation, the derived altimeter setting for the current position of the aircraft.

3. Obtain the altitude (from MSL) of both stations and the aircraft from an exponentional function, and then subtract those altitudes to obtain Mode C Altitude.

RSS uses all stations within a distance of 500 nm to derive an altimeter setting. It uses distance weighted square average to interpolate station pressure observations.

Example:

Assume an airplane flying at 10,000 feet above MSL at 3 degrees latitude and 2 degrees longitude. Three stations are located around the airplane. The locations, elevations from MSL and observed air pressures of these stations are in the following table.

	Latitude (degree)	Longitude (degree)	Elevation (feet)	Observed Air Pressure (inch Hg)
Station 1	0	0	1000	28.92
Station 2	5	5	5000	24.2
Station 3	0	5	3000	26.9

Table 1: This table provides the data inputs used for the above example.

For interpolation between these three stations, the weighted square average is used. The factors in this average are the inverse square of the distance. The interpolation of elevation and pressure is as follows:

$$average\_pressure = \frac{28.29/d1^2 + 24.2/d2^2 + 26.9/d3^2}{1/d1^2 + 1/d2^2 + 1/d3^2}$$

where d is the ground distance between stations and the airplane.

d1 = 400.9255 km

d2 = 400.2916 km

d3 = 471.7263 km

By substituting d, the average-pressure is 26.6474 inch Hg. With the same procedure, the average-elevation is 3002.32 feet. The following function is used to obtain the altitude (above MSL) from the air pressure:

$$altitude = (1 - (\frac{pressure}{altimeter\_setting})^{0.190284}) \times 145366.45$$

The unit for altitude is feet and for pressure is inches Hg.

By substituting the average-elevation=3002.32 feet and average-pressure=26.6474 inch Hg, in the above formula, the altimeter setting would be 29.736.

The Transponder Actor has five input ports. Ports P11-P13 are the output ports of the Event Analyzer Actor, which extracts data that is about Mode A or Mode C Altitude and directs them to the Transponder Actor. The P14 input-port is for triggering the Transponder Actor to produce its output and P17 input-port provides the aircraft's altitude to calculate Mode C Altitude. Figure 18 shows the Transponder Actor and its sub-actors.

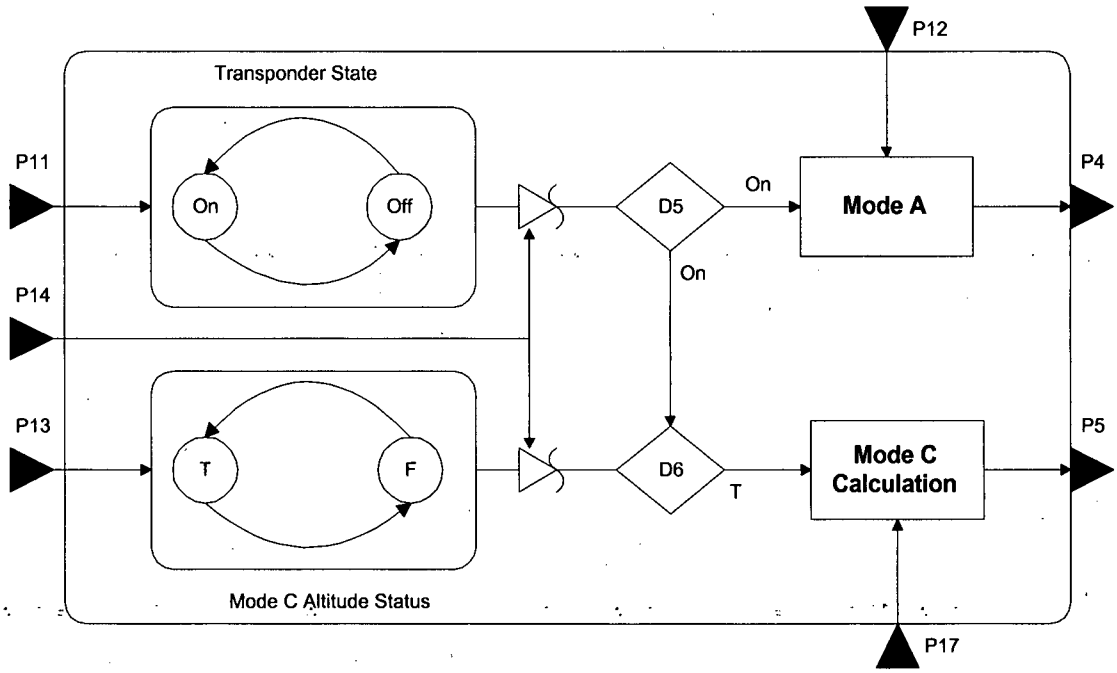

Figure 18: Transponder Actor

The Transponder Actor includes two finite-state machines. One of these machines specifies whether the transponder is 'on' or 'off' and the other one determines the status of the transponder in response to the interrogation Mode C Altitude. The results of these state machines go through decision-makers D5 and D6, which pass the results if the states are 'on.' Like the Aircraft Locator Actor the Transponder Actor is associated with an input-port for an incoming trigger. The trigger activates the actor to generate its output.

3.4.7 Trigger Actor

The Trigger Actor defines the time-steps at which the Aircraft Actor must produce its output. The length of a time-step, which is the update-time for an aircraft, is important from the viewpoint of performance. A suitable update-time prevents unnecessary operations from updating the location of the aircraft which consequently improves the efficiency of the system.

Towards finding a suitable update-time for an aircraft, our modelling approach partitions the set of aircraft actively being simulated at any instant of time into two groups. The first group contains aircraft that are in radar range (in-sight aircraft). The second group contains aircraft that are not in radar range (out-of-sight aircraft). For out-of-sight aircraft the update-time of 60 seconds is chosen. The out-of-sight aircraft changes to in-sight aircraft whenever it enters into a specific distance of radar range.

For in-sight aircraft if we want to have one observation in each radar rotation, the suitable update-time can be derived from radar rotation rate, e.g., if the radar rotation rate is 15 rpm the suitable update-time would be 4 seconds. This choice puts limits on the RSS to function properly.

For the RSS, another solution has been suggested. This solution requires that we put an upper limit on the aircraft's groundspeed. For example, the maximum groundspeed for an aircraft is 900 km/h. This limit in groundspeed is required to have RSS's radar detect aircraft as they do in the physical world. Otherwise, RSS may miss detecting an aircraft in one turn or detect it in a wrong location. To better understanding why a speed limit is imposed, consider the following scenario.

Suppose that radar area is divided into 180 sectors, the rotation rate is 5 rmp (each rotation takes 12 seconds) and the distance between the aircraft and the radar is short (about 10 km). The radar detects the aircraft at sector 1 in second 1. In the next 11 seconds the real aircraft moves to sector 180 (because the distance is short, the sectors are small) but at time 12 (when the radar is pointing to sector 180) the radar does not detect the aircraft at this sector because the location of the aircraft has not yet been updated. At time 13 (when the radar is pointing to sector 1) the aircraft updates itself and the new location will be in sector 180 so the radar does not detect the aircraft again.

Now if the maximum aircraft speed is known it is possible to derive with minimal time that is needed for an aircraft to change its sector, so a better update-time can be chosen.

For example in above scenario an update-time of 11 seconds (or less) is chosen instead of 12 seconds.

To find a suitable update-time four factors are considered:

1. Radar rotation rate
2. Distance between the aircraft and radar source
3. Radar beam width
4. Aircraft's speed

The update-time for in-sight aircraft would vary between 1 second (the assumed minimum update-time) and the radar's rotation time. The optimum update-time in seconds is calculated from the following formula:

$$update\_time(second) = \frac{dis(m) \times beam\_width(degree) \times \pi \times 3.6}{180 \times speed(km/h)}$$

Example:

Consider an aircraft flying at a speed of 500 km/h with a distance of 55 km from the radar source. The radar is rotating at 10 rpm with beam width of 3 degrees. In this case the optimum update-time would be:

$$update\_time(second) = \frac{55000 \times 3 \times \pi \times 3.6}{180 \times 500} = 20.73$$

Since update-time is greater than rotation time, which is 10 seconds, so the update-time would be 10 seconds. But if the distance of the aircraft decreases to 10 km the update-time would be 3 seconds.

$$update\_time(second) = \frac{10000 \times 3 \times \pi \times 3.6}{180 \times 500} = 3.77$$

The radar triggers all aircraft to update their location first. After the situation between the aircraft and radar is found, the aircraft is responsible to initiate an event to update itself.

In all, the recommended solution to re-calculate the position of an aircraft depends on dynamically partitioning all actively simulated aircraft into "within radar range" and "outside radar range". This partitioning is used to avoid unnecessary computation by limiting frequent updating to the "within radar range" group of aircraft. However, the cost of this efficiency measure is the creation of a data flow from the environment back to the aircraft model. This data flow does not correspond to any physical interaction. It is purely a computational artefact.

3.5 Radar Modelling

The RSS simulates the physical and dynamic behaviour of the ATC surveillance radar, PSR and SSR in a manner that models the rotating radar antenna and the operation of the radar in target detection. The simulator considers some of the physical and dynamic characteristics of surveillance radar, which define the state variables of radar objects. These characteristics are as follows:

1. Radar source location: determines the location of the radar site by its longitude and latitude.

2. Radar elevation: determines the height of radar antenna from mean sea level (MSL).

3. Modality: specifies the kind of surveillance radar that is active. A radar source can have one active PSR, one active SSR, both of them or none of them. Modality is the only dynamic characteristic of radar source, which can change during a simulation.

4. Maximum angle of elevation: determines the maximum vertical angle of an emitted energy beam by a radar antenna.

5. Maximum slant range: determines the maximum distance between an aircraft and the radar so that the radar can detect the aircraft.

6. Radar beam width: specifies the angle on the horizontal plane subtended by the beam of the radar antenna.

7. Number of beam positions: specifies the number of "beam positions" in one full rotation of the radar antenna. ("Beam position" is one of the fixed positions in the rotation of radar at the horizontal plane at which the radar may receive the radar reply.)

8. Number of rotations per minute: determines the number of rotations that the radar antenna makes in one minute.

The radar transmits its energy impulses in a discrete manner, at every beam position.

3.5.1 Radar Transmitter Actor

The Radar Transmitter Actor is modeled as a discrete-event component, as its beam position varies discretely with time. It is a self-firing actor that generates events to trigger itself for each update.

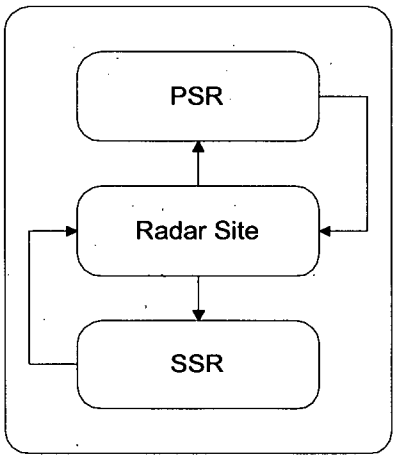

Figure 19: Radar Transmitter Actor

The Radar Transmitter Actor is a composite actor, which contains three other actors. These actors represent the Radar Site, PSR and SSR.

3.5.1.1 Radar Site Actor

The Radar Site Actor encapsulates the common characteristics of PSR and SSR, which is information about site location and site elevation. This information is released through ports P3-P5 upon receiving a trigger through port P1. The modality of the radar site can be changed upon receiving a trigger from port P2. Knowing the modality of the radar site, the Radar Site Actor sends messages to PSR and SSR actors through ports P6 and P7 to enable or disable their operation. The decision-maker D0 decides whether to send messages to the PSR or to the SSR Actor.

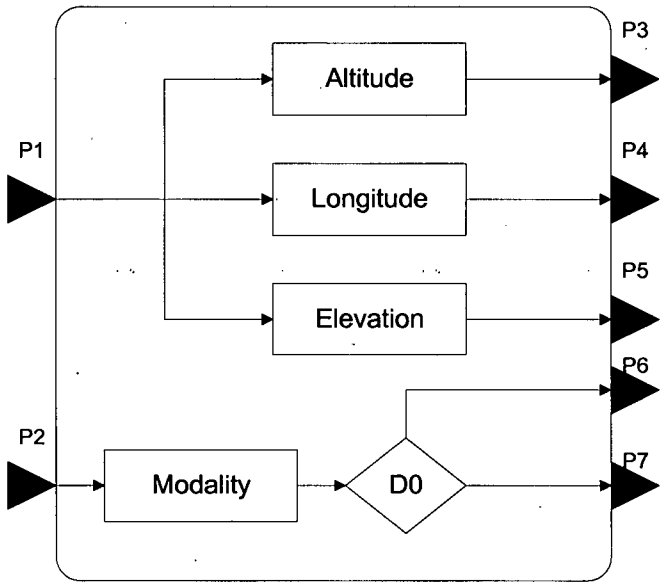

Figure 20: Radar Site Actor

3.5.1.2 PSR and SSR Actors

At the second level of the hierarchy, the PSR and SSR actors have the same composition. So here, we only describe the composition of the PSR Actor.

The PSR Actor consists of a state machine that specifies whether the PSR is enabled or not. This state is changed upon receiving a message from P9. The PSR antenna position gets updated by receiving a trigger from port P8. After setting the new position, if the radar state is 'on', the PSR Actor releases radar data through ports P10-P13. This data is released by functions: Horizontal Angle, Max Slant Range, Max Angle of Elevation and Beam Width.

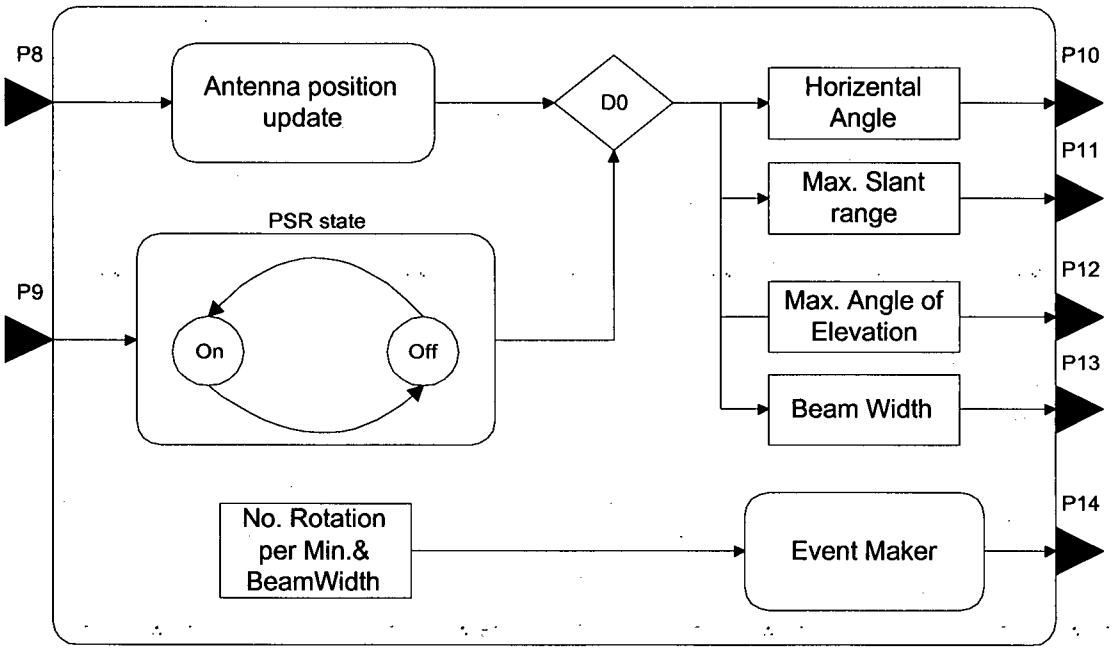

Figure 21: PSR Actor

The Event Maker Actor from the radar composite actor makes event repeatedly and sends it to the event queue. The function "No. Rotation per Min& Beam Width" triggers the Event Maker Actor to make those events. This event is fired to the Radar Actor at its time of execution. The frequency of generating this event is:

$$F = \frac{6 \times No.ofRotationsPerMin}{BeamWidth}$$

Upon receiving the event via port P8, the Antenna Position Update Actor calculates the new horizontal angle of the radar antenna and sets the antenna in the new position.

$$AntennaNewPosition = \Pr eviousPosition + beamWidth$$

3.5.2 Radar Receiver Actor

The Radar Receiver Actor collects radar replies and combines them to produce the radar returns. A radar return consists of a concert of information about the detected aircraft

such as location, altitude and identification code. If there is any reply that cannot be correlated with the other replies, the reply is sent to the output directly to get processed by the Radar Data Processing System.

The following figure shows a Radar Receiver Actor and its composition:

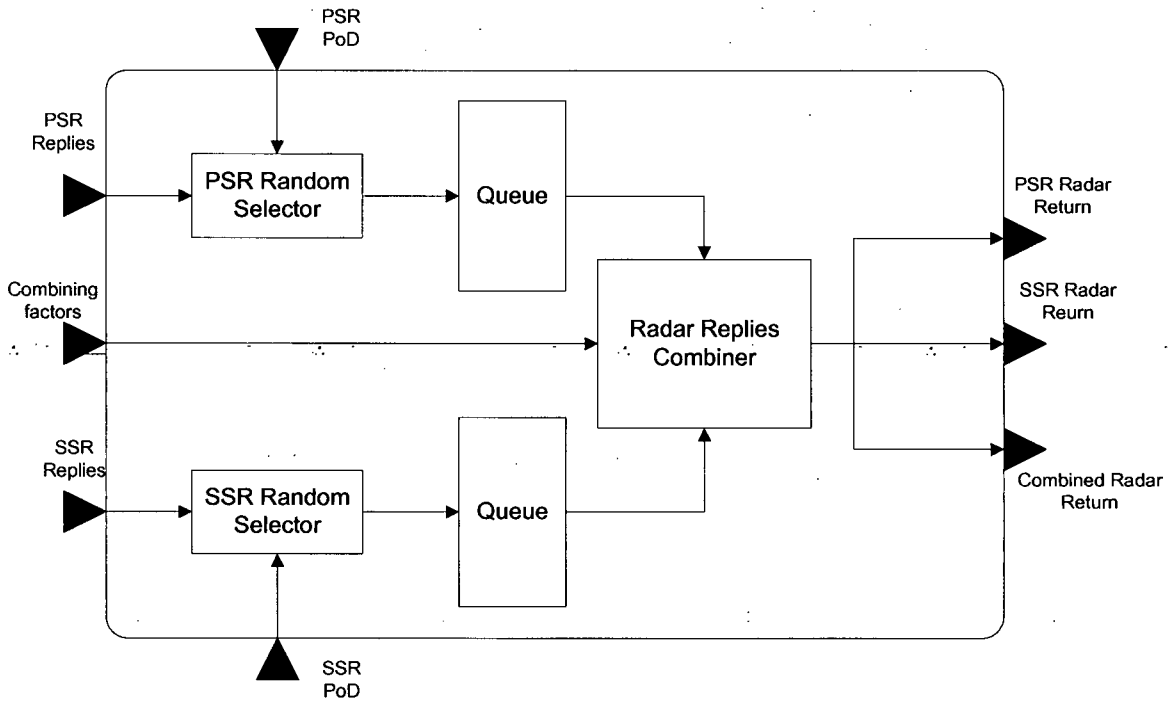

Figure 22: Radar Receiver Actor

The actor has two input ports from which it acquires radar replies from the Environment Actor. The replies go through Random Selectors, which pick up some of the replies randomly and discard the rest. The Random Selectors take into account the probability of target detection and simulates the event that some signals and interrogations would remain without replies. For example, if a SSR sends out 100 interrogations only 80 interrogations are responded to by transponders and 20 of them would be ignored by transponders, obstructed in transmission or distorted. The refined replies are sent to queues for storage. The Random Selectors have input ports to obtain the radar probability of target detection. This probability depends on the number of radar beam positions, location radar site, radar type and probability of transponder response for the SSR.

Queues are for storing replies for one radar rotation. After the end of each rotation, the replies are sent to the "Radar Replies Combiner" to get merged and combined. The "Radar Replies Combiner" separates radar replies according to various correlation factors and combines that data to produce radar returns. The input port P0 provides the factors that are used for the reply combination. For example, these factors can define the range-azimuth zones, which are used to verify whether two replies are related to one target or not. The output of the actor could be in form of PSR return, SSR return or a combined PSR/SSR return.

3.6 Environment Modelling

The RSS simulates the environment that fulfils the required conditions in which PSR or SSR can detect a target. The environment serves as an intermediary between radar and aircraft. In particular, it carries both transmitted and reflected electromagnetic energy between radar and aircraft that can be affected by terrain, weather and environmental factors. Its behaviour is more complicated than a simple messenger or conduit between these two actors. The environment modifies the "data" exchanged between radar and aircraft. For example, the environment causes the direction of the radar beam to change slightly due to refraction.

The conditions for radar detection are as follows:

1. The radar is currently activated.

2. The slant range is not greater than the radar maximum slant range.

3. The line of sight is between the horizon (i.e., the tangent line at the location of the radar source) and the maximum radar elevation.

4. The azimuth is within the radar beam width for the current beam position of the radar antenna.

5. The line of sight does not intersect a surface representing an obstruction.

These conditions are modeled as a series of decisions makers in PSR and SSR Environment Actors. If all of these conditions are satisfied, the radar can detect the target; otherwise, the target is invisible to the radar.

The composite Environment Actor is composed of one PSR Environment Actor and one SSR Environment Actor, which present the required conditions for target detection by PSR and SSR respectively.

The input of the PSR Environment Actor and the SSR Environment Actor comes from the Aircraft Actor, the Radar Actor and the Obstruction Actor.

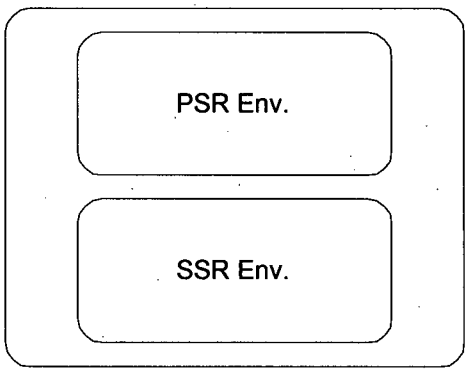

Figure 23: Environment Actor

3.6.1 PSR Environment Actor

The PSR Environment Actor contains several operational functions and decision makers. These decision makers are ordered from the least computational expensive function to the most expensive. This ordering is designed to increase the efficiency of the simulation model by avoiding more computationally expensive calculations than necessary in a particular situation. The actor proceeds with checking the next condition only if the previous condition is satisfied.

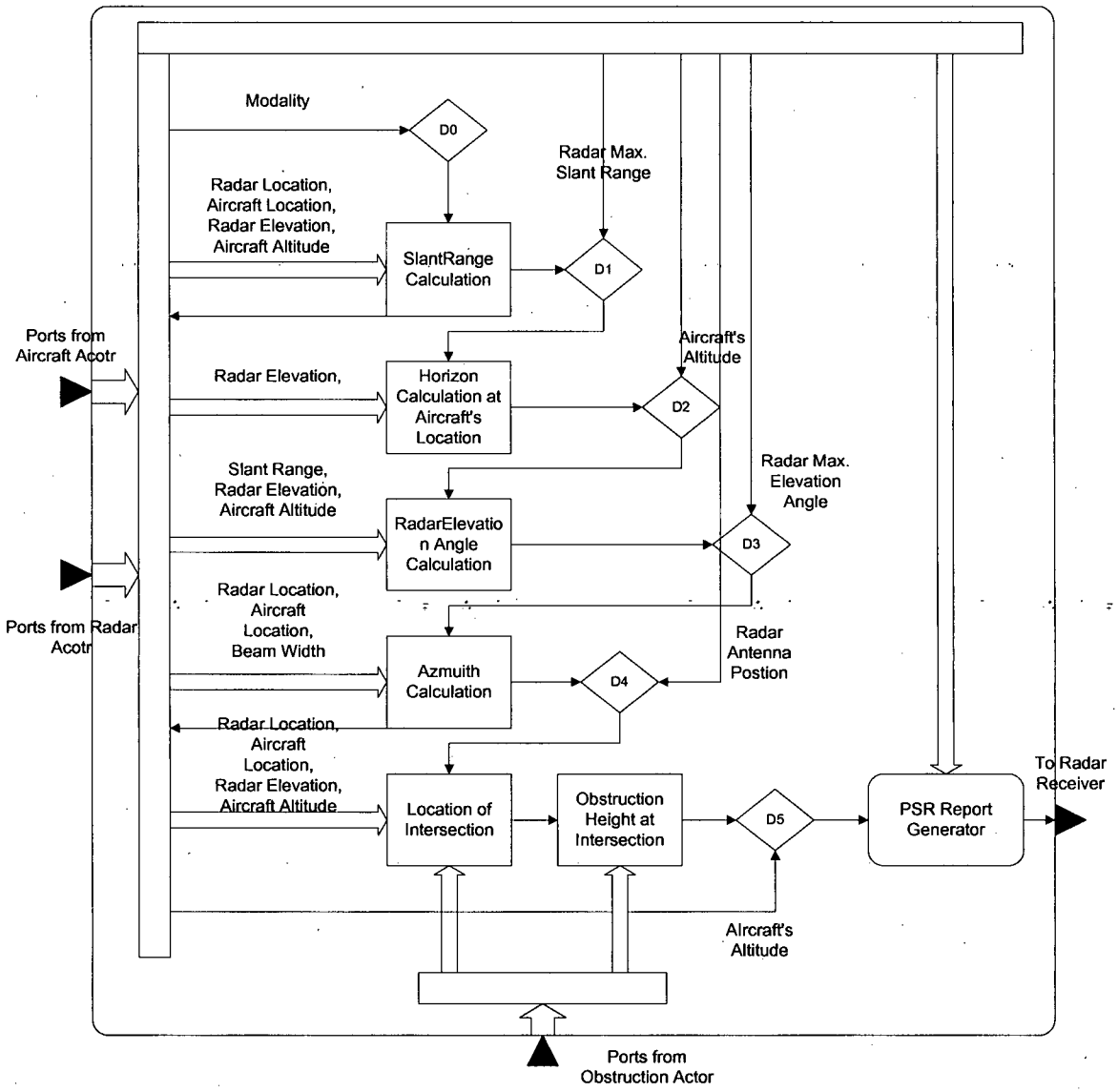

Figure 24: PSR Environment Actor

The first decision maker, D0, reflects the first condition for the PSR to detect an aircraft. If the modality of the radar site indicates that the PSR is in operation, the PSR Environment Actor starts verifying the second conditions.

For the second condition, the actor first calculates the slant range between the aircraft and the PSR. If the slant range is less than the PSR Maximum Slant Range, the actor moves on to verify the next condition.

In the third stage, the actor calculates the height of the radar horizon line at the aircraft's location and compares it to the aircraft's altitude.

In the fourth stage, the actor computes the Radar Elevation Angle at which the target is being detected and compares it to the Maximum Radar Elevation Angle.

In the fifth stage, the actor calculates the azimuth at which the target is being detected and verifies whether the difference between the radar antenna position and the azimuth is within the radar Beam Width or not.

Finally, the actor finds out the intersection of the line of sight, the imaginary line between the target and the radar, with every obstruction, and verifies whether the line is above the obstruction, at the intersection point, or not.

If all the conditions are satisfied then the PSR Report Generator Actor generates a PSR report for the detected aircraft. The generated report is sent to the Radar Transmitter Actor.

3.6.2 SSR Environment Actor

The operation and composition of the SSR Environment Actor is similar to the PSR Environment Actor with this difference that SSR Environment Actor has a different SSR Report Generator Actor to generate reports about the aircraft's identification code and altitude.

3.6.3 Slant Range Computation

Obtaining the slant range is not straightforward because radar signals experience a small amount of refraction as they pass through the atmosphere. The reason is that the atmosphere is a heterogeneous medium and its refraction index varies with height. The

refraction bends the signals towards the ground so that their range is beyond the physical horizon.

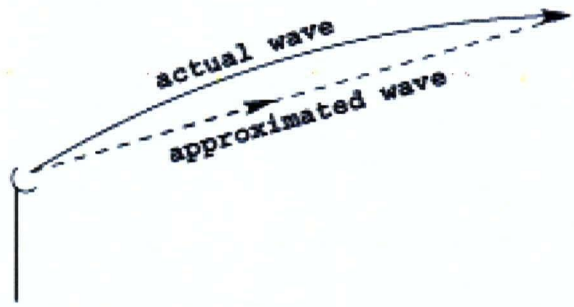

Figure 25: Actual Wave Propagation

The effect of refraction on radar signals is advantageous because it allows the radar to "see over the horizon." (The path of the ray starting horizontally from the radar is called radar horizon.) Tracing the radar paths is rather complex. It is more common to simplify this process and presume a fictitious radius for the earth to apply the simple computation for the radar horizon. In this case it is presumed that the atmosphere is homogenous and does not bend the radar rays.

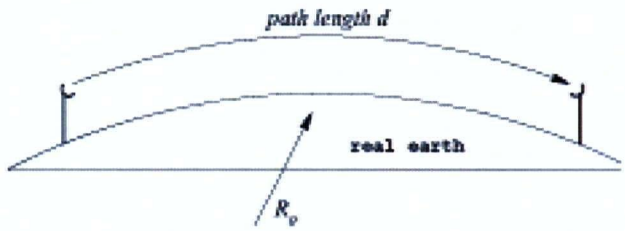

Figure 26: Real earth and wave path

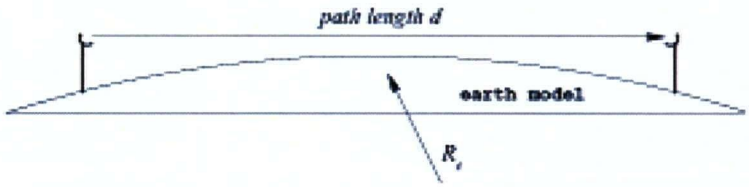

Figure 27: Earth model and wave path

This fictitious radius is 4/3 times the true value of the earth's radius and is equivalent to the assumption of a linear decrease in the refraction index with a height of 3.95*10-5 per kilometer.

The more accurate estimation for refractive index is using an exponentional function. The following function describes the behaviour of the refractive index with height.

$$n(h) = 1 + ae^{-bh}$$
$$a = 315 \times 10^{-6}$$
$$b = 0.316 \times 10^{-3}$$

This expression gives a more accurate refractive index of the dry atmosphere with respect to height; however, water vapour and some other atmospheric conditions may complicate the matter and require numerical computation to evaluate the refraction index.

The Refraction model from Doviak and Zrinc in 1992 [26] shows that the four-thirds model gives over 99% accuracy of the radar horizon for a high angle of elevation, but this accuracy comes down by decreasing the angle of elevation and increasing the distance of the aircraft.

Refraction Models from Doviak and Zrnic (1992)

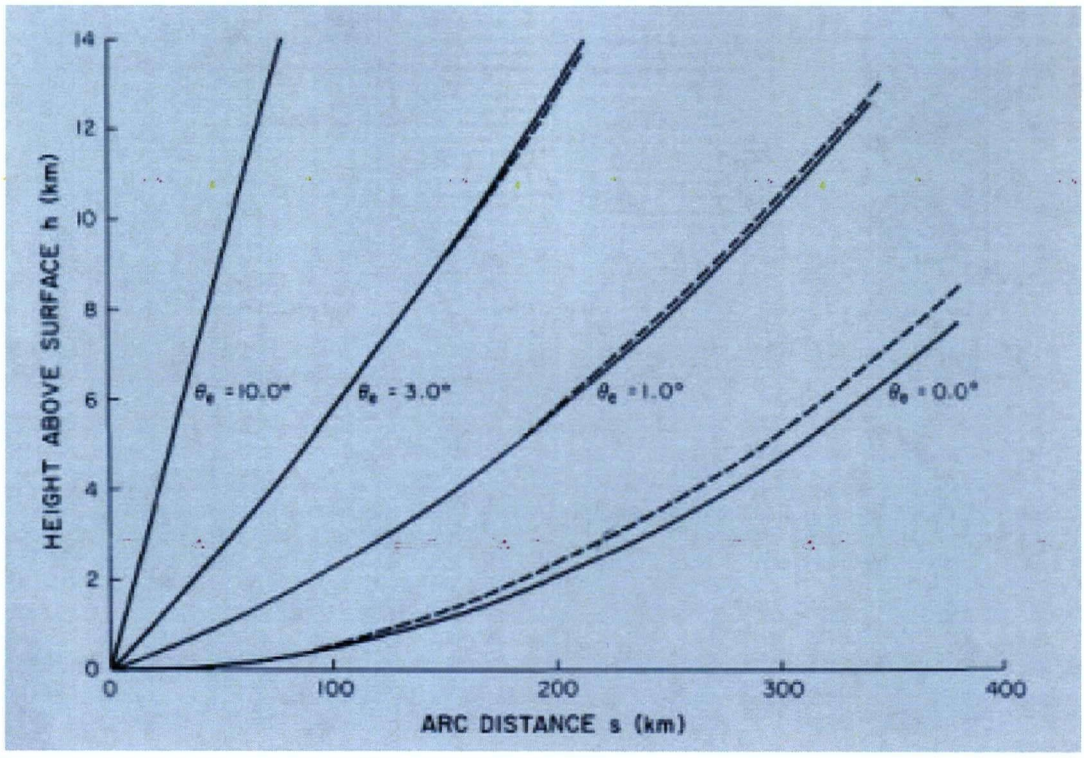

Figure 28: The accuracy of four-thirds model decreases by increasing elevation angle.

Graphs of the Four-thirds Model show that for low angles of elevation that typically all SSR and PSR radars apply (the typical value is about 0.5 degree) and that the radar horizon is little higher than its true height. This difference increases with the distance of the aircraft but it is not counted as an important factor when comparing a simulation with reality. In reality, different weather conditions may impose adverse factors, such as causing subrefraction, supperfraction and trapping. Even this inaccuracy between the Four-thirds Model and reality is taken into consideration. It is applied to far-off aircraft, over 300 kilometres away. At this distance there are numerous additional issues to consider that are of greater importance than this inaccuracy. For example, most of the ATC radars do not detect such distant targets and neither do transponders detect distant radar sources.

Since the Four-thirds Model is sufficiently accurate and simple to use, the RSS makes use of this model for estimating the radar horizon.

3.7 Future Enhancement on the RSS model

The RSS has two significant parts, the simulation of aircraft trajectory and the simulation of the radar. The simulation of aircraft trajectories is relatively simple mathematical model of objects moving in space as determined by initial position, direction, speed and elapsed time. The second of these parts uses the aircraft trajectory to generate radar returns. A very simplistic simulation model for ATC radar might simply be a function that samples the aircraft trajectory at regular intervals. In contrast, the model presented in this chapter takes accounts of several important "real world" effects that contribute complexity and a degree of uncertainty to the relationship between aircraft trajectory and positional information reported by ATC radars. We now briefly consider some additional enhancements that could be added to our model to more fully take account of this complex relationship.

The problem with the simulation of the radar is that the simulation considers one reply for a transmitted signal or pulse, while in the physical world a radar may receive extraneous replies for the following reasons:

1. Fruit replies: fruit is the term given to transponder replies when they are not relevant to the interrogations. Most of these replies are received by antenna side-lobes. To understand the events that result in fruit, assume there are two radar sources, A and B, and two aircraft, C and D. Radar A interrogates aircraft C just after radar B sends its interrogation. The result is that radar A receives two replies, one for its own interrogation and the other for the interrogation from radar C. The other event happens when aircraft C is replying to radar A and aircraft D is replying to the radar B, therefore both radar sources receives two replies from both of the aircraft.

2. Residual hybrids: Hybrids occur when side-lobe fruit reply interacts with an overlapping main beam reply. There are various possible overlapping situations that end in hybrids.

3. False replies due to reflection and multipaths: radar replies may reflect from objects in the air or on the ground and cause extraneous replies at the radar station. Reflection is the most significant cause of overlapped replies. It produces multiple paths between an aircraft and the radar. Besides, it can produce virtual multi-transponder for one aircraft. These multiple paths can produce delayed echoes of each pulse that may be seen as separate replies. The multiple paths can be produced by reflection from earth and other such obstructions. The second event that is caused by reflection is virtual multi-transponder. This type of error happens when an aircraft reflects the reply of another aircraft so that it has appears to have more than one transponder.

4. Wide pulse replies (transponder error): wide pulse replies are caused two ways:
   a. The transponder may be out of collaboration and transmit wide replies,
   b. Multiple paths can produce delayed echoes of each pulse, which may be seen as additional replies.

While the simulation model described in this chapter already accounts for some of the "real world" complexity of the relationship between aircraft trajectory and positional information derived from ATC radar returns, the incorporation of the above mentioned enhancements would improve the realism of the simulation model . For example, in the simulation of radar using a variable beam width for PSR and omni-directional radar as complementary radar with SSR would help in considering the effects of side-lobes.

3.8 Summary

This chapter describes a model to simulate the ATC surveillance radar system. The model is based on the actor model that supports hybrid systems and hierarchy. The model consists of Aircraft, Radar, Obstruction and Environment Actor.

The Environment Actor is a key actor that connects a Radar Actor to an Aircraft Actor. Using an Environment Actor as a mediator between other actors separates actors from each other. This causes the system to be modifiable and reusable. The Environment Actor also simulates the environmental factors that influence the signal transmission between aircraft and radar.

The Aircraft Actor is a self-firing actor that updates its state. The interval at which the actor updates its states is variable. This variable timing makes the system efficient as it removes unnecessary re-calculation.

The Radar Actor is responsible for simulating the operation of PSR and SSR. The frequency at which the radar sends its signals depends on its beam width and frequency of rotation rather than the number of beam positions. This abstraction avoids explicit simulation of tens of interrogations or signals per second and consequently improves the efficiency of the simulation.

4 RSS Software Architecture

4.1 Introduction

This chapter provides a comprehensive architectural overview of the RSS system. It uses a number of different architectural views to depict different aspects of the system and to capture and convey significant architectural decisions. It also describes some technological and product factors that are considered in the architecture of the system.

In Chapter 3, a model for the system was introduced. The model explained the different types of objects in the system and the connections between them. The model is described at a level of abstraction where the objects concurrently exist in an active state ready to respond to stimuli. This chapter concerns itself with the implementation of this model as a software system. The implementation of this model must also provide mechanisms to support the various capabilities of objects such as object creation and communication between objects.

Here, in the architecture of the system, we also explain the RSS simulation engine. The RSS simulation engine is responsible to create and dispose of objects. It establishes communication paths between objects. It defines what creates objects, and how and when objects are created. It defines how and when objects communicate with each other and when and how objects produce results.

4.2 Architecture Representation

We use Kruchten's 4+1 model [25], as a guide for the architectural description of the RSS software architecture. In particular, our description includes a logical view, a process view, a development view and two scenarios to support these views. Although Kruchten's 4+1 model also calls for a physical view, we have not created a physical view for our description of the RSS architecture since the hardware platform used to support

the execution of the RSS software is generic and has not influenced any decisions about the software architecture.

4.3 Technology Reuse and Developing Environment

The system is developed using C++ for execution on a single machine with the UNIX operating system. The primary reason for choosing C++ is the simple fact that the author has much more experience in using this language than other general purpose programming languages. Use of the class mechanism in C++ facilitates the translation of the RSS implementation into other object-oriented languages such as Java and C#.

The system uses Lex and Yacc for parsing the input scripted commands. It also uses the C++ STL (Standard Template Library) to benefit from previously implemented abstractions for data containers, such as Set, Vector, and Map.

4.4 From Model to the Architecture

Chapter 3 explained that the RSS model is based on the concept of discrete-event simulation. The components in discrete-event models typically use events to communicate to each other. The simulation engine has a global event queue, which sorts events by their time-stamps. When a component generates an output event, the event is placed in the queue. At each iteration of the simulation, the event with the earliest time-stamp is taken from the front of the queue and its destination component is executed [18]. Here in the RSS, we take the same approach to design the simulation engine.

The RSS model described earlier in Chapter 3 identifies the components of the system and the way that they are connected to each other. These components are mapped one-to-one to classes, which are described in the logical view. In the logical view, we also describe the role of some additional classes, which create the runtime platform for the system. Next, the process view describes the independent execution threads of the system and finally the development view organizes the classes into layers and explains the

dependency between the layers. The architecture of the RSS must also meet other factors such as modifiability, usability, and integrability.

4.5 Product Factors

There are four factors that are considered in the architecture of the RSS: the performance, modifiability, integrability and usability of the system.

4.5.1 Performance

The performance of the RSS has two aspects: queue size and latency. The queue size defines the maximum number of objects that the RSS can handle during the simulation. By using the STL Data Containers, the number of simulated objects is limited by the system's available memory. The following table illustrates the estimated size of each simulated object in the RSS. The data requirement for each object is provided by the RSS requirement specification.

Aircraft	Radar	Station	Obstruction
200 Bytes	150 Bytes	60 Bytes	55 Bytes

Table 1: Example of memory usage by each RSS object

Within the capacity of a general purpose computing platform to support an application that requires several megabytes of memory for data storage, the RSS can hold a larger number of objects, for example the data of 5000 aircraft require 1 MB memory. Therefore, the only concern with restricting the queue size in RSS is that the more objects results into the more system latency.

In the RSS, the latency of the simulation is made using two methods: the Aircraft Update Method and the Radar Seek Method. The Aircraft Update Method is to update the location of each aircraft. The Radar Seek Method is used by radar to search for detectable aircraft. During each search the radar examines all the aircraft to find out whether they are detectable or not. The latency of the system depends on the number of calls for these

methods. Next, we describe the factors that are considered to optimize the number of calls for these methods.

4.5.1.1 Factors Considered in Aircraft Update Algorithm

As mentioned in Chapter 3, the update-time for each aircraft depends on its distance from the radar sources, the radar beam width and the speed of the aircraft. The algorithm avoids unnecessary computation to update the location of the aircraft.

4.5.1.2 Factors Considered in Radar Seek Algorithm

The frequency of calls for the Radar Seek Method is defined by the number of radar beam positions. Therefore, the number of beam positions has a significant effect on the performance of the simulation.

As mentioned before, surveillance radars have hundreds of beam positions. This large number of beam positions is required to obtain a sufficient number of radar replies to increase the Target Detection Probability Rate (TDPR) and the accuracy of the azimuth and range measurements. In other words, The Target Detection Probability Rate reflects the number of beam positions. For example, a high Target Detection Probability Rate indicates high number of beam positions. Hence, instead of simulating radar with a different number of beam positions, the system defines a minimum number of beam positions and then simulates the effect of having more beam positions by changing the Target Detection Probability Rate. The minimum number of beam positions is calculated by dividing the whole circle by the radar beam width. For example, for a beam width of 2 degrees the number of required beam positions is 180.

4.6 Modifiability

The RSS system should be modifiable. Changes may be made to add new simulated objects, new environmental factors, modify the user interface, or add a new interface

compatible to other external systems. Changes may also be made to modify the way that the environment object connects two modeling objects.

4.7 Integrability

The RSS is part of a radar simulation tool, so it is very likely that it connects to other subsystems. For this reason the RSS must be designed in such a way that it can be integrated with other subsystems easily.

4.8 Usability

The RSS must provide facilities for the user to analyze different scenarios. Therefore, the RSS must be designed so that the user can conveniently review the simulation results without interfering with the simulation process. It is also desired that the RSS would accept changes to the scenario while the simulation is running. This way, the user can observe the results of changes in the input without undergoing the whole simulation. For example, the user can interactively change the aircraft's route or add a new radar site to the system.

4.9 Logical View

The logical view of the RSS architecture describes the RSS classes and their organization in the RSS subsystems. The following class diagram illustrates the relationships between the classes and the subsystems of the RSS.

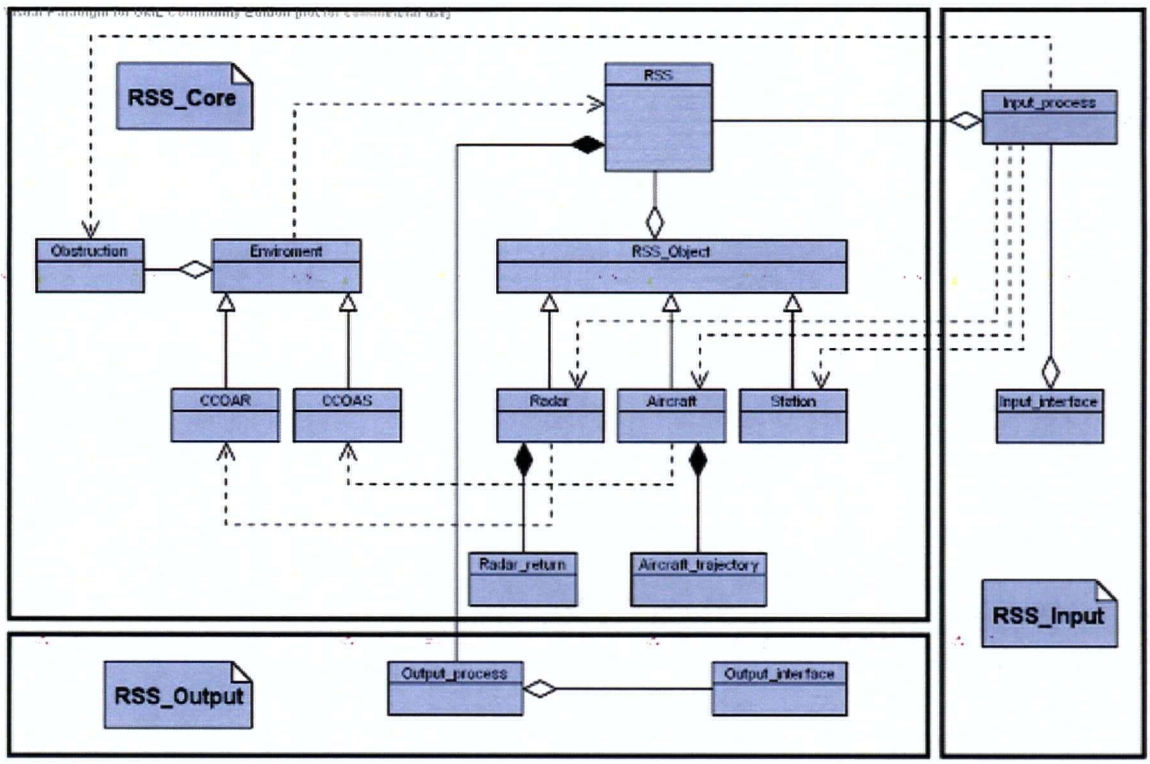

Figure 29: RSS Class Diagram, containing the RSS three subsystems

The RSS system consists of three subsystems:

1. RSS_Input: This subsystem includes two classes: the Input_interface and the Input_process. The RSS object creates one object from each of these classes in runtime. The Input_interface object aggregates the Input_process object. The Input_interface object implements a proper interface to take data from the user. The Input_process object converts the input data into events and modeling objects, which are fed into the Core_RSS subsystem. The RSS_Input subsystem provides the RSS with the ability to read different types of input by accepting different interfaces from external systems. This architectural approach provides the RSS with integrability.

2. Core_RSS: This is the main subsystem of the RSS. This part of the system performs the simulation by processing events. Each event is associated with a modeling object that encapsulates the procedures and methods that perform the simulation.

The main class of this subsystem is the RSS class. The RSS object launches the simulation and contains the data containers used to hold objects that are being simulated. The RSS object is unique in the simulation and is associated with one Input_process object and one Output_process object. The Singleton pattern is used to allow having just one instance of the RSS, the Input_process and the Output_process classes.

Following initialization, the RSS object starts retrieving the events from the event queue. These events are the results of the RSS_Input operations and are sorted according to the timestamp of an event in the event queue. After retrieval, the RSS finds the related modeling object for an event and sends the event to the modeling object as a message for execution. The modeling objects are provided by the RSS_Object subclasses, such as the aircraft class and the radar class. The RSS waits until the execution of one event is finished and it proceeds by retrieving another. Execution of each event may produce other events, modify a status of a modeling object or produce simulation results. Each modeling object handles its own simulation results. The simulation results are in form of either the Radar_return object or the Aircraft_trajectory object. Then the simulation results are sent to the RSS_Output subsystem. The RSS has three following modeling classes, which are inherited from the RSS_Object class.

1. The Radar class, which models the operation of a radar site.

2. The Station, which holds the station pressure observations. These observations model the aircraft's altimeter in response to the interrogation Mode C.

3. The Aircraft, which models movement of the aircraft.

The RSS uses the Environment objects to establish the connection between the modeling objects. The Environment class provides a unified interface to different types of environmental objects such as obstruction or weather. The RSS has only one

type of environmental object, which is the Obstruction object. The way that RSS can handle different environmental objects is supported by the Façade pattern. By the Façade pattern, the Environment class is considered as a unified interface that connects the RSS into other environmental classes. For example, if we add a class to implement weather, the Environment class would aggregate the new Weather class. This approach addresses modifiability for the RSS to accept new objects.

The Environment objects play a very significant role in the RSS. Not only do they connect modeling objects together, but they also define the principles and rules of connections between the modeling objects. For example, they define the conditions in which the radar can detect an aircraft. A factor that is considered here is that these principles must be decoupled from the objects. This decoupling facilitates any future modification of these rules. The Environment class is an abstract class from which two classes are inherited. These classes, which are the CCOAR class and the CCOAS class, impose the rules and principles of connecting modeling objects. A CCOAR object connects an Aircraft object to a Radar object. A CCOAS object connects an Aircraft object to a Station object. The way that different modeling objects are connected to each other through an Environment object is supported by the Mediator pattern.

3. RSS_Output: This part of the system processes the results of the simulation into a proper form to display. This subsystem includes two main classes, the Output_process class and the Output_interface class. The Output_process object is for processing the simulation results. The Output_interface object is for displaying the results.

   The Output_process class is associated with the RSS class to receive simulation results in the form of Radar_return objects and Aircraft_trajectory objects. The Output_process object processes the results and sends them to the Output_interface object to display. The Output_interface class also implements the methods that are

necessary to map simulated entities on the screen and provides a proper interface to show the simulation results.

4.10 Process View

The RSS has three processes that each runs on their own thread of execution. These three processes are mapped one-to-one into three subsystems of the RSS, which are described in the logical view.

The following table illustrates a brief description of these processes.

Name	Trigger	Implementation	Mapping subsystem
Data_In	User	Obtain data from the user through an interface and convert them into events and modeling objects	RSS_Input
Core_RSS	Timer	Process events by executing different methods corresponding to different events and obtain simulation results	Core_RSS
Data_Out	Timer or Core_RSS	Obtain simulation results from the Core_RSS subsystem and convert them into a proper form to display, and provide an interface to display the results	RSS_Output

Table 2: The RSS's three processes

The Data_In process: This process contains two major tasks. The first task is provided by methods in the Input_interface class that implement the user interface for entering

data. The second task is provided by the Input_Process class that converts input data into series of events and modeling objects. The trigger of this process is an event from the user that is handled by Input_interface object. This process runs as a separate thread so that the user can enter data without interfering with the simulation. This process uses the same resources as the Core_RSS process does. These resources are data containers that hold events and modeling Objects. So, a synchronizing method is used between these two threads to avoid invalid data access.

The Core_RSS process: This is the main process that handles the simulation. In this process the execution control is managed by the RSS object that fetches events from the event queue and executes them. Regarding different events, the RSS object sends the events to different modeling objects. After processing the events, the simulation results are stored to feed other RSS subsystems. The trigger for this process is time. Whenever the local RSS clock reaches the timestamp of an event, that event is executed. The Core_RSS process shares two resources with two other processes, the simulation result containers and the data containers. The results of this process appear as the Aircraft_trajectory and the Radar_return objects that are saved in the simulation result containers in the Output_process object.

The Data_Out process: This process contains two major groups of procedures. The first group is provided by the Output_process class, which converts simulation results into a proper form of data. The second group is provided by the Output_interface class that represents a suitable user interface to display the simulation results. The trigger of this process is time. The clock that is generating this time is handled by the Output_interface object so that the user can play the simulation at any time. This process runs on its own thread of execution so that the user can analyse the output and play the results over and over without interfering with the simulation. Since this process uses simulation results as its input, changing the display-time by the user does not require re-simulation. This approach makes the system usable for the user in terms of having a comfortable way to analyze the simulation results and having an interactive simulation. The Data_Out process shares simulation result containers with the Core_RSS process.

4.11 Development View

The development view describes the modules of the RSS. The RSS consists of six modules:

1. Object_Hiding module
2. Process_Hiding module
3. RSS_Output_Process module
4. RSS_Input_Process module
5. RSS_Input_Interface module
6. RSS_Ouput_Interface module

The following figure represents the development organization of these six modules in four layers.

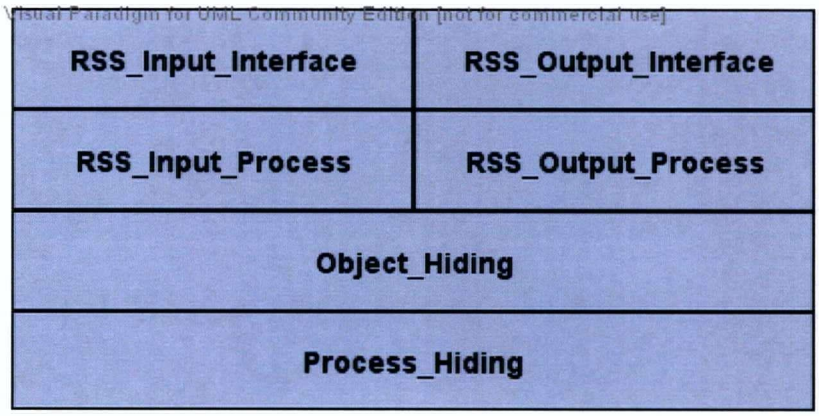

Figure 30: The 4 layers of the RSS

RSS_Ouput_Interface and RSS_Input_Interface Modules

These modules contain classes that need changing if the RSS input/output user interfaces change. The classes of these modules implement the RSS user interface and provide facilities to obtain data from the user and to display the RSS results.

RSS_Input_Process Module

This module hides the RSS from the input user interface. This module contains classes that need changing if the user interface changes or if a new user interface is added to the system. This module converts the format of the input data, obtained by the input user interface, into a suitable form for the RSS.

RSS_Output_Process Module

This module hides the RSS from the output user interface. It contains classes that need changing if the output user interface changes or if a new user interface is added to the system. This module converts the format of the simulation results into a form suitable for the RSS output user interface.

Object Hiding Module

This module contains classes that require changes if a new simulated entity is added to the system. This module hides the details of the RSS modeling and environmental objects and carries the common concepts and abstractions of all objects that are simulated by the application.

Process Hiding Module

This module contains classes, which carry behaviours and specifications of different simulated objects. This module hides the detailed implementation of each simulated object and decouples the object implementation from its abstraction.

The following table illustrates the mapping of classes into these modules.

Module name	**Mapping classes**
RSS_Input_Interface	Input_interface
RSS_Ouput_Interface	Output_interface
RSS_Input_Process	Input_process
RSS_Input_Process	Output_process

Object_Hiding	RSS, RSS_Object, Environment
Process_Hiding	Radar, Aircraft, Station, Obstruction

Table 3: RSS's five layers of architecture

4.12 Scenario

The following scenario is used to demonstrate interactions between the RSS objects and the messages that are transferred between them. The collaboration diagram is used to illustrate the objects and their interactions.

In this scenario there are two aircraft and a single radar source. The first aircraft is BC727 initiated at time 00:00:00 at -112/59.5 Longitude/Latitude and flying to the east at the of speed 350 nm. The second aircraft is AH01 initiated at time 00:10:00 at -112/59.3 Longitude/Latitude and flying to the east at the of speed 250 nm. The radar site starts as a PSR radar from time 00:00:00 and changes its modality to PSR and SSR at time 00:14:33 which results in having both PSR and SSR returns from time 00:14:33. The simulation runs for 20 minutes. The output of the system is reports about the position of the aircraft and radar returns which include the azimuth and the range of detected aircraft. The summary of the scenario which includes a selective number of radar returns in the output column is described as follows:

Time	***Input***	***Output***
	Radar site KTX at -110/60 Longitude/Latitude elevation 2000 feet from MSL PSR max slant range is 100 nm PSR max elevation angle 20 degrees PSR beam width 3 degrees	(The radar is initiated here with modality of combined. From now on the radar searches for the airplane.)

	PSR rotation rate: 10 rounds/min PSR number of positions: 300 SSR max slant range: 220 nm SSR max elevation angle: 30 degrees SSR beam width: 2 degrees SSR rotation rate: 15 round/min SSR number of positions: 400	
00:00:00	PSR active only	(From now on only PSR part of radar site works)

00:00:00	Aircraft BC727
	-112/59.5 Longitude/Latitude
	speed 350 nautical miles
	bearing (direction) 90 degrees
	altitude 20,000 feet
	with Transponder on
		`aircraft bc727 at (59.500,-112.000)`
		`bearing = 90`
		`ground speed = 300`
		`altitude = 20000`
		`transponder status = on`
		`mode A SSR code = 1200`
		`mode C altitude status = on`
		`mode C altitude = 199`
		
		(Every 30 seconds the system generates a report for this airplane)

00:00:04	-	`radar ktx`
		`range = 67.781 nm`
		`azimuth = 244.34 degrees`
		`modality = PSR_only`
		`radar part = PSR`
		
		(The first time that radar observes the airplane; the airplane is observed approximately every 6 seconds.)

00:07:30	-	aircraft bc727 at (59.500,-110.778) bearing = 90 ground speed = 300 altitude = 20000 transponder status = on mode A SSR code = 1200 mode C altitude status = on mode C altitude = 199
00:07:57	-	radar ktx range = 37.186 nm azimuth = 215.66 degrees modality = PSR_only radar part = PSR
00:10:00	Aircraft AH01 -112/59.3 Longitude/Latitude speed 250 nautical miles bearing (direction) 90 degrees altitude 5000 feet with Transponder off	aircraft AH01 at (59.300,-112.000) bearing = 90 ground speed = 350 altitude = 20000 transponder status = of mode A SSR code = 1200 mode C altitude status = on mode C altitude = 199
00:10:03	-	radar ktx range = 32.327 nm azimuth = 200.19 degrees modality = PSR_only radar part = PSR radar ktx range = 74.154 nm azimuth = 236.02 degrees modality = PSR_only radar part = PSR
00:14:33	PSR and SSR both active	status of radar ktx changed to ***combined***

00:14:33	-	radar ktx range = 54.903 nm azimuth = 219.94 degrees modality = combined radar part = PSR
00:14:34	-	radar ktx range = 32.455 nm azimuth = 159.18 degrees modality = combined mode A SSR code = 1200 mode C altitude status = on aircraft name bc727 radar part = SSR
00:14:35	-	radar ktx range = 54.776 nm azimuth = 219.78 degrees modality = combined mode A SSR code = NULL mode C altitude status = off aircraft name NULL radar part = SSR
00:14:38	-	radar ktx range = 32.570 nm azimuth = 158.63 degrees modality = combined radar part = PSR
00:20:00	Ends Simulation	-

Table 4: Scenario

The following collaboration diagram shows the objects in the RSS_Input subsystem for this scenario.

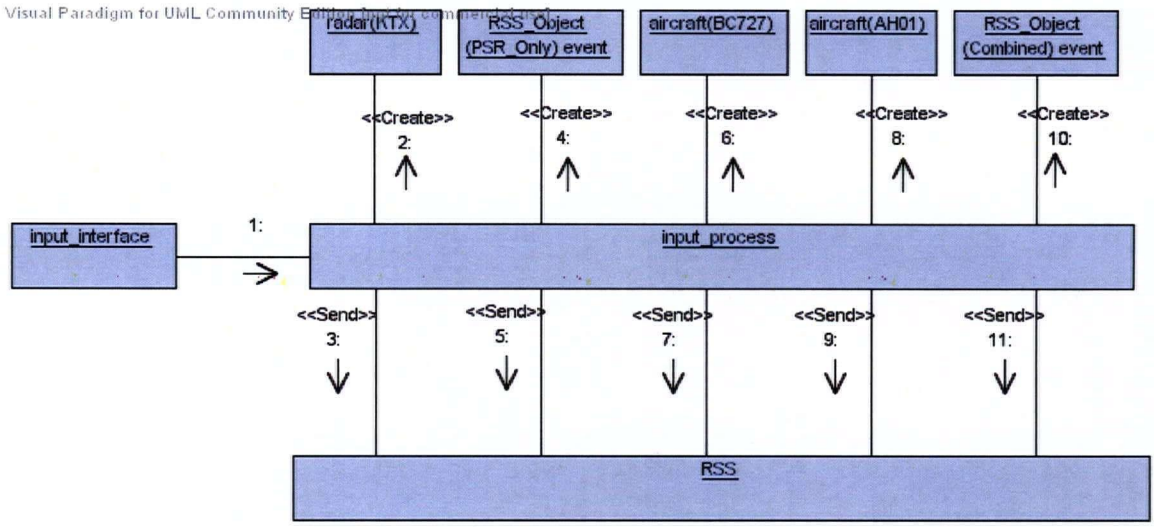

Figure 31: Collaboration diagram, objects participating in the RSS_Input subsystem

As illustrated, the Input_interface object obtains the input commands. Then it sends the attributes of the command to the Input process object. Since the command is about initiating a radar source, the Input_process object creates a radar object named KTX and sends it to the RSS object. Confronting the command to change the modality of the radar source to PSR, the Input_process object creates a RSS_Object object and loads it with required data and then sends it to the RSS object. Interactions from 6 to 11 are for creating other two aircraft and an event to change the modality of the radar source.

The Input_process class has association with modeling classes and the RSS-Object class. The RSS receives the first input command as an RSS_Object object and holds it in its event queue. The received Radar object goes to the data container for future use.

The following diagram illustrates the collaboration between objects in Core_RSS subsystem.

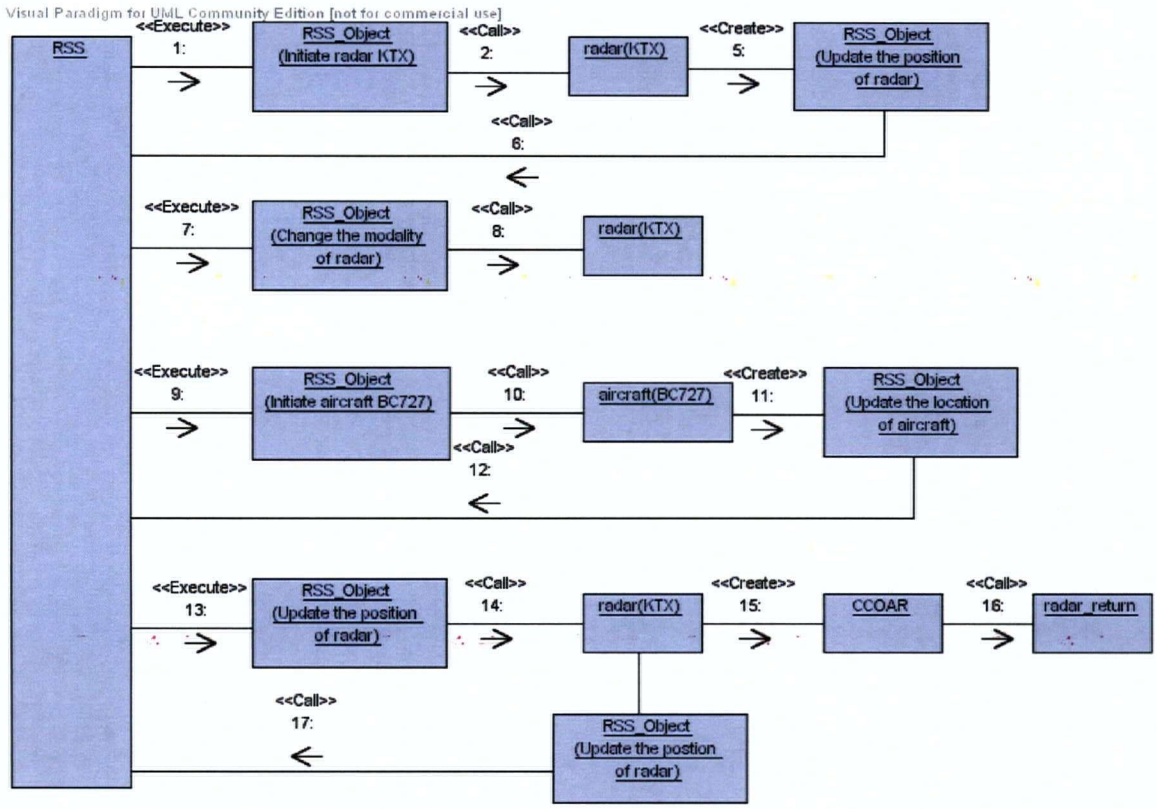

Figure 32: Collaboration Diagram, Core_RSS subsystem

In Core_RSS subsystem, the RSS object retrieves the first event (presented by RSS_Object object) from the event queue. The RSS object executes the event. Upon being executed, the RSS_Object invokes the corresponding method in the related radar object. The radar object executes the method to initiate the radar source. After that, the radar object creates an RSS_Object to update its own position in the future. At collaboration 13, after execution of the event which updates the radar position, the Radar object creates a CCOAR object. The CCOAR object searches for aircraft that can be detected by the radar, considering the existing environment. The object that establishes connection between the Radar object, Obstruction objects and Aircraft objects is a CCORS object. The results of radar detection are sent to the Radar_return object. Then, the Radar_return object is sent to RSS_output subsystem for display purposes.

The diagram below illustrates the collaboration between objects in RSS_Output subsystem.

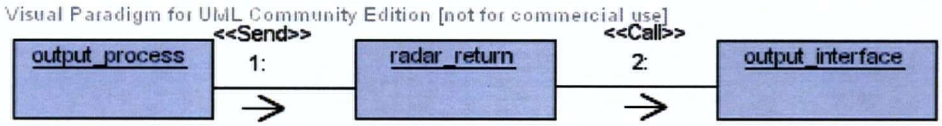

Figure 33: Collaboration Diagram in the RSS_Output subsystem

In the RSS_Output subsystem the Output_process object reads outputs from its result containers. The results of RSS_Core subsystem are in the form of either a Radar_return or an Aircraft_trajectory object that are stored in result containers of the Output_process object. The Output_process object performs necessary processes on the Radar_return or the Aircraft_trajectory object. Later on, these objects call related functions in the Output_interface class to show the simulation results.

4.13 Summary

This chapter explains the architecture decisions that are made for the RSS architecture. Building upon the RSS model described earlier in Chapter 3, the architecture of the RSS indicates that the components are using message-based communication to exchange data. It also shows that the runtime system of the RSS uses the event queue to manage communication between the components.

This chapter describes how the system performance is achieved by using variable update-time and the Target Detection Probability Rate to decrease the latency of the system.

Our description of the RSS architecture is based on Kruchten's 4+1 model. The logical view of the system describes the additional classes that must be considered for the runtime system. It also depicts more detail of the association between the classes and the subsystems of the RSS and how the system can achieve integrability and modifiability. The process view explains the processes of the system and how the system can achieve

usability. Finally, the development view of the architecture shows that the RSS classes fall into four layers. Each layer indicates how a change in a class may affect other classes.

5 RSS Testing

5.1 Introduction

This chapter describes the validation and verification process of the RSS model and the RSS simulation system. It explains different techniques to test the RSS simulation system. It uses an object-oriented testing approach to perform unit testing on RSS. The chapter also explains the testing methods that are used to test significant functions of RSS.

The results of RSS are complex and it is difficult for humans to verify the accuracy of RSS's outputs. In other words, the accuracy of the results cannot be measured easily by the tester. It is true that to some extent the tester can determine if the results are completely unreasonable, but even if the results are in the range of correct answer then deciding whether the test results are absolutely correct is still difficult. For example, it is difficult and time-consuming to calculate the final location of an aircraft after flying for one hour and having a number of turns.

5.2 Verification and Validation of simulation models and systems

Simulation models and systems are increasingly being used to solve problems and make decisions. The developers of these models essentially follow a model development process. In the model development process, the developers first have a problem entity which is an idea, situation or phenomena to be modeled. Having the problem entity, they make a conceptual model which is a logic/mathematical representation of the problem entity. Then based on the conceptual model, they define a design and computerized model of the simulation application. The computerized model is developed through a computer programming and implementation phase [29].

The verification and validation of a simulation model is over all phases of the model development process. Model verification deals with building the model right and model validation deals with building the right model. The conceptual model must be valid. The theories and assumptions underlying it must be correct. The computerized model must be also validated and verified. It must be a right implementation of the conceptual model and its operation has sufficient accuracy.

The challenge with testing a simulation system is that project managers often underestimate the amount of effort required to verify a simulation tool. As well, to verify the behaviour of a simulation it is required to know what the correct behaviour is under the precise conditions being used to test the simulation tool. It may be very difficult and sometimes even impossible to determine the exact details of this correct behaviour. When it is not possible to determine whether the result of a simulation is correct, the verification and validation process is used to build the user's confidence in the simulation system. To obtain the user's confidence, the simulation system must give accurate outputs for the most probable inputs from the user. In other words, the system must be valid for the scenarios that are sufficiently characterize the behaviour of the simulation that the application will use most [29].

Some of the validation and verification techniques are [30]:

1. Comparison to other valid models.
2. Event validity, in which the events of simulation model are compared to those of the real system.
3. Face validity, which is asking experts about the validity of the system.
4. Animation, in which the system's behaviour is displayed graphically.
5. Predictive validation, which is used to predict the system's behaviour and then comparisons are made between the real system and the model.

5.3 RSS Testing

RSS testing was performed during several cycles in modelling process. The RSS model gradually constructed and validated by people who had knowledge of the problem domain. Based on the RSS model, over several cycles, the computerized model is gradually built. At the end of each cycle, some of the validation and verification techniques are used to test the implemented model.

RSS testing uses bottom-up testing strategy. The testing is implemented in five phases. The first three phases (RSS_Input testing, RSS_Output testing and RSS_Core testing) are to test three subsystems of RSS. Each of these three phases tests one of the subsystems of RSS and includes unit testing and integration testing. The fourth phase is the integration testing of the whole RSS system and the fifth phase is RSS system testing.

In RSS testing some validation and verification techniques such as animation, comparison to other models, event validity, face validity and predictive validation are used.

5.4 RSS Unit Testing

Unit testing is the lowest level of formal test activity. In the structured programming a procedure, function or module may be treated as a unit for testing purposes. In Object Oriented Programming (OOP), a class is the unit of development and therefore of testing [26]. Unit testing on OOP cannot easily be mapped onto the testing of individual object's operations. Its behaviour is meaningless unless analyzed in relation to other operations and their joint effect on shared data. Therefore, any significant unit typically cannot be smaller than one class [27].

One of the characteristics of object-oriented software is the complex dependency that may exist between classes due to inheritance, association and aggregation [27]. This complexity draws attention to the importance of having a test order to minimize testing

efforts. For example, a child class is better tested after its parent class because it is affected by any changes in its parent class.

The object-oriented testing approach that is taken for RSS testing procedures is defined in the article "Testing Levels for Object-oriented Software" [27]. The approach takes into account complex static and dynamic (polymorphism) dependencies between classes in OOP. It represents a test order by graphing which testing levels must be done in sequence and which ones may be done independently. It also provides information about the classes involved in each level of testing.

Unit testing is performed in three phases of RSS testing, RSS_Input testing, RSS_Output testing and RSS_Core testing. The testing of each subsystem of RSS starts with unit testing and ends with integration testing of the subsystem.

5.5 Testing Issues on Radar System Simulators

Testing the accuracy of some key functions complicates the unit testing of an application such as RSS. These functions in RSS are as follows:

1. The function to simulate the movement of an aircraft
2. The function to simulate the effects of obstructions
3. The function to simulate the action of radar to detect an aircraft

These functions are the most important functions of the system and the accuracy of the whole simulation depends on the accuracy of these functions. To test the function that simulates the movement of an aircraft, we acquired assistance from another simulation tool, namely, Matlab toolbox. The Matlab toolbox program receives some input data about the location of the aircraft and the distance that is traveled. Then by having the aircraft's initial location and the traveled distance, the systems generate the final location of the aircraft. The same input data is fed into the RSS, using some methods to read the file and write the results into the file. By comparing the results, we could measure the

accuracy of the RSS system in comparison with a reliable simulation tool such as Matlab toolbox.

The influence of having obstructions in the simulation is verified through some scenarios. By assumption, the radar is unable to detect an aircraft if an obstruction is present. Therefore, some scenarios were made in which the radar could detect an aircraft. Then we modify the scenario with an additional obstruction in the path of the aircraft and radar. We tested the simulation of an obstruction by using different heights and sizes for the obstruction in different locations in the path of the aircraft and radar.

To verify the function that simulates the detection of an aircraft by radar, we compare the radar returns for an aircraft with the simulation's results of the aircraft movements. Two issues are important in this matter. The first is that the radar must submit one return for each rotation and the other one is that the radar range, the azimuth and the latitude of the aircraft must give the same location for the aircraft as the positioning method. Details on this calculation were described earlier in Chapter 3.

5.6 RSS Subsystem Testing

As mentioned before, the RSS system is divided into three subsystems. Therefore, the testing of the RSS is accomplished by testing these subsystems. The order that the classes are tested is important in each subsystem.

5.6.1 RSS_Input testing

In the testing of the RSS_Input subsystem, the Input_Process class of RSS_Input subsystem is first tested. The parts of the Input_Process class that are related to the RSS class are replaced by stub methods. Stubs methods simulate the correct answer of the real methods. After that, the Input_Interface class is tested in association with the Input_Process class.

5.6.2 RSS_Output testing

In the testing of the RSS_Output subsystem, the Output_Process class of RSS_Output subsystem is tested to verify that it implements correctly. The parts of the Output_Process class that are related to RSS class are replaced by stub methods. After that, the Output_Interface class is tested in association with the Output_Process class.

5.6.3 RSS_Core Testing

In order to find the test order in this subsystem, the Kung Algorithm [27] is used. This algorithm is based on Class Diagram and called Object Relation Diagram (ORD). The ORD captures the static dependencies between the classes. The nodes in ORD represent the classes and the edges represent the relationship between the classes. For any two classes C1 and C2:

1. An edge labelled I from C1 to C2 indicates that C1 is a child of C2.
2. An edge labelled Ag from C1 to C2 indicates that C1 is an aggregated class of C2.
3. An edge labelled As from C1 to C2 indicates that C1 is associated with C2.

The following graph represents the ORD of the RSS_Core subsystem.

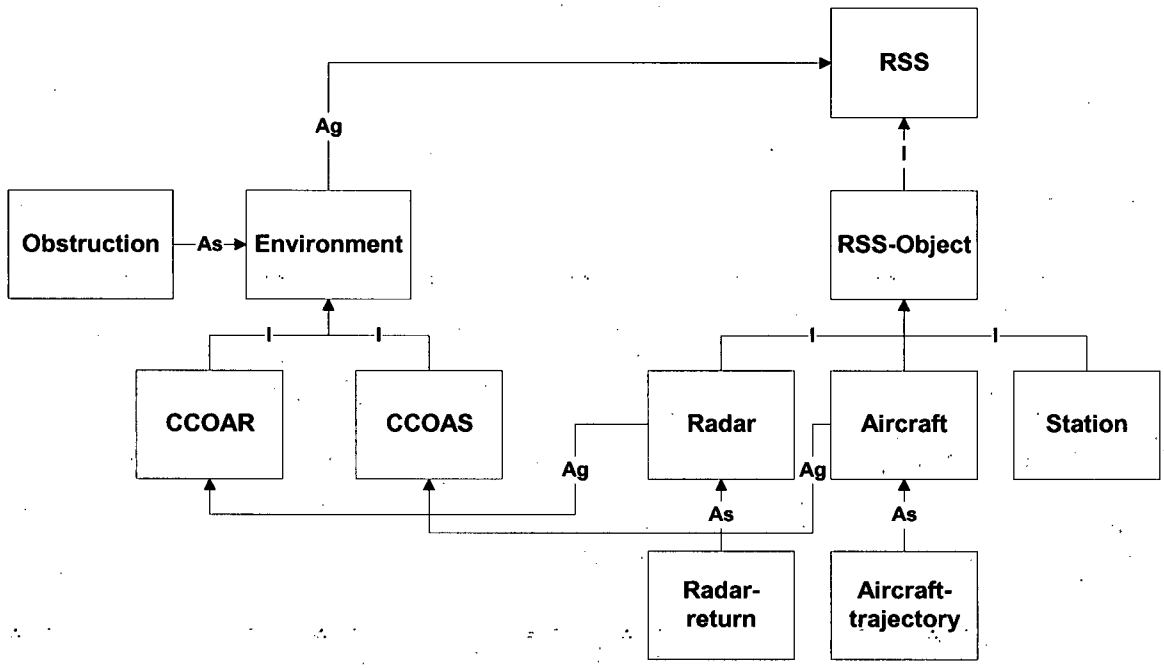

Figure 34: ORD of RSS_Core subsystem

The main idea of the algorithm is to test more independent classes first to minimize the test effort. The following table shows the test order for the RSS_Core subsystem based on this algorithm.

Testing level	Classes
1	RSS
2	Environment, RSS-Object
3	Obstruction, Station, CCOAR, CCOAS
4	Radar, Aircraft
5	Radar-return, Aircraft-trajectory

Table 5: Testing levels for RSS's classes based on static relationship

The above test order is based on the static relationships between the classes of the RSS_Core subsystem. This order may be changed for a dynamic relationship. The dynamic relationships between classes occur at the execution time. For example, the

Environment class associates with the RSS class, which is a parent class of the Radar class. Due to polymorphism, the Environment class may dynamically interact with the Radar class and thus should be tested after the Environment class.

In covering the dynamic dependencies between the classes, the second test order is defined. This test order is based on dynamic relationships between the classes of the RSS_Core and is implemented after the first test order, which is defined based on static dependencies.

The dynamic relationship between class A and B occurs when class A is statically associated with parent classes of class B. The following graph represents the dynamic relationship between the classes of the RSS_Core subsystem.

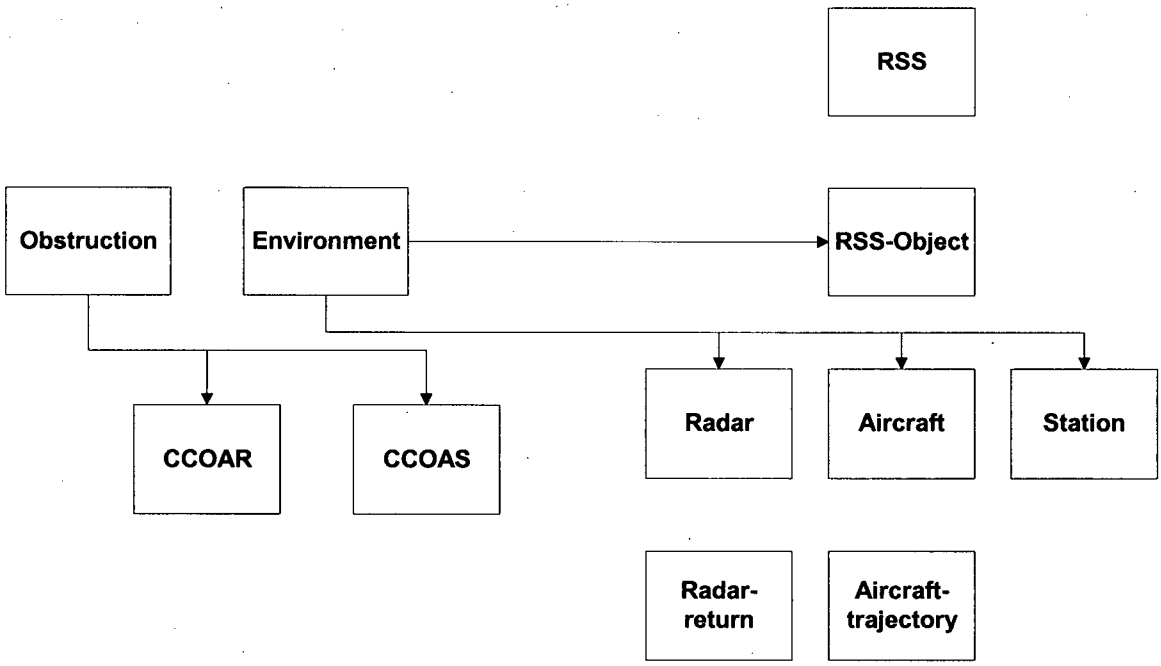

Figure 35: Dynamic relationship between classes of RSS_Core subsystem

According to this dynamic relationship, the other test order is defined. The following table shows this test order.

Testing level	Classes
1	Radar, Aircraft, Station, CCOAR, CCOAS, RSS-Object
2	Environment, Obstruction

Table 6: Testing levels for RSS's classes based on dynamic relationship

After testing individual classes, the integration testing is performed based on the static and dynamic interactions between classes. The following table reveals the test order for the purpose of integration testing.

Testing Level	Class	Dependent classes
1	RSS	-
2	RSS-Object	RSS
3	Environment	RSS
4	Station	RSS-Object
5	CCOAR	Environment
6	CCOAS	Environment
7	Obstruction	Environment
8	Radar	RSS-Object, CCOAR
9	Aircraft	RSS-Object, CCOAS
10	Radar-return	Radar
11	Aircraft-trajectory	Aircraft
12	Environment	Radar, Aircraft, Station, RSS-Object
13	Obstruction	CCOAR, CCOAS

Table 7:Testing levels for RSS's classes

This test order is derived from tables 1 and 2. As illustrated in the table the Environment and Obstruction classes must be tested again at the end of the testing procedure because of dynamic dependencies that may have occurred in runtime.

To test RSS_Core subsystem we perform several testing techniques:

1. Event validity: The subsystem is tested by monitoring the size of event queue and the events that are produced by each object. For example, aircraft are supposed to produce events at specific intervals and these events must be added to the event queue with their time-stamp and the last location information of the aircraft.

2. Comparison to other systems: As mentioned before, some methods of the aircraft object and radar object such as the method to simulate the movement of the aircraft are tested by comparing to some other valid simulation applications such as Matlab.

3. Predict validation: Since the system is observable, meaning that data of particular variables can be recorded and observed as the simulation is running; it is possible to modify some input data and observe the behaviour of the system against this modification.

5.7 RSS Integration Testing

Testing the integration of RSS's three subsystems requires the verification of the consistency between the interfaces and concurrency between the processes. The concurrency between processes is tested by using a method which indicate data and a process that tries to access to it. The other part of integration testing is to test the data transaction between subsystems in runtime. For this purpose, stub methods are used to represent the outgoing incoming data as text files. By comparing text files we can verify if each piece of data is transferred between subsystems.

5.8 RSS System Testing

The model of the RSS has been also verified and validated by a knowledgeable person. RSS system testing is verified to meet its Software Requirement Specifications (SRS).

The test is implemented by a group of graduate students. The result of these tests indicates that RSS satisfies its requirement specifications. In addition, the models' operational behaviour is displayed graphically as the simulation is running. The movement of the aircraft and displayed radar plots show that the simulation behaviour is reasonable.

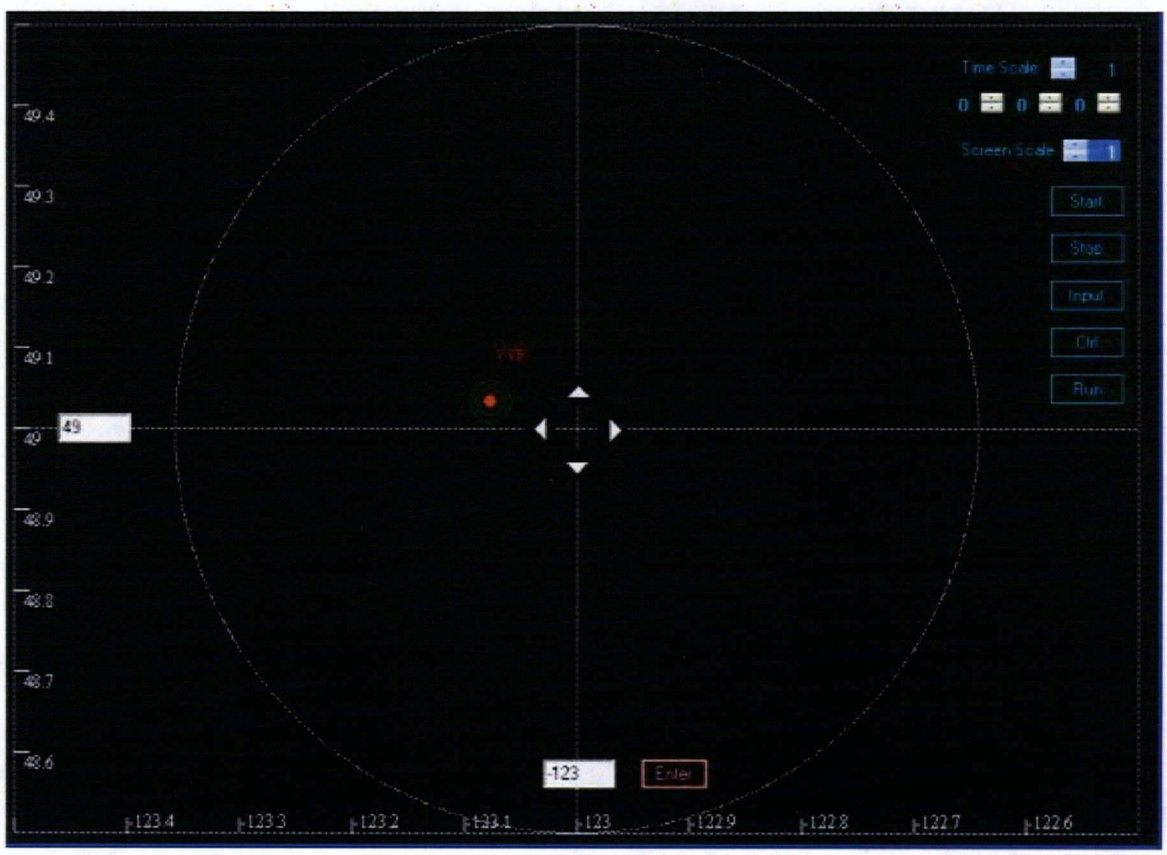

Figure 36: The animated result of the simulation system.

5.9 Conclusion

The testing efforts of the simulation application such as RSS must focus on testing the accuracy of its significant functions. In RSS, these functions are methods to simulate aircraft movements, obstructions and radar. The testing activity for RSS includes different techniques of testing the accuracy of these functions. In addition to these tests, the testing effort can be minimized and made efficient if it is performed based on the dependencies of the classes.

6 Conclusion

6.1 Conclusion

This thesis satisfies the technological needs in designing and developing software applications to simulate target detection with radar. The thesis gathers the information of a surveillance radar system. This information is necessary to manipulate the model and design of ATC radar simulators. It contains details of radar systems from radar antenna to the Plot Extraction Unit.

In this research a model for a computer simulation application is described which involves a target, a means of detection, such as radar, and the environment between the radar and the target. The environment element is a key aspect of our model, and encompasses some of the more innovative aspects of our research. It imposes the rules and policies to connect radar to the target; it defines when a target is visible to the radar. The benefit of using the environment element as a mediator between the radar and the target is that it decouples the implementation of the target and the radar. In addition, the environment element can apply the influence of other elements such as weather on the relationship between the radar and the target. This makes the environment flexible so as to accommodate other rules and principles of connecting the radar and the target. This model provides modifiability and extensibility to the system.

In addition to modifiability and extensibility of the system, the model also addresses the performance of RSS. From a performance point of view, the model considers the most ideal time to update the location of a target and the number of interrogations that are sent by the radar source. The model measures the time intervals to update the location of a target from the situation of the radar and the target. This technique removes unnecessary re-calculation and improves the performance of the system. In addition to updating the location, the model calculates the number of radar interrogations based on the frequency of rotation instead of on the number of beam positions. Although the number of beam

positions is defined as an input for the radar system, it does not mean that the simulation must simulate this number of beam positions. The simulation finds the ideal number of beam positions so as to have the top performance. It also considers the effect of the entered number of beam positions as a factor in defining the probability of target detection. The decision prevents the radar from sending tens of interrogations (for SSR) or signals (for PSR) per second and consequently improves the efficiency of the system.

The software architecture of RSS takes into account other requirements of RSS such as usability and integrability. Composed of three subsystems, the architecture of RSS simplifies the integration of RSS to other external systems. RSS consists of three processes that separate the interaction of the user with the system and the simulation process. The benefit of using RSS architecture for a similar system is that it uses the same architecture pattern which follows the RSS's model and uses well-known design patterns such Mediator and Singleton.

The RSS simulation model is extensible and modifiable. The RSS simulation model considers the possible extension in simulation system in case that simulation of some other objects; which may affect the environment between aircraft and radar may add to the system. It also considers that simulation methods may change by time as some more accurate and efficient simulation methods are created. In addition, the RSS model provides the ability to RSS to have different environment of aircraft and radar. In other words, the environment between one aircraft and radar can be different from the environment between another aircraft and radar. This characteristic enables the simulation model to perform simulation according to the situation of aircraft and radar which enhances the performance of the system. Extensibility, modifiability, efficiency and realism are the important characteristics of the RSS simulation model.

Biography

[1] Drik L.Kettenis, "*An Algorithm for Parallel Combined Continuous and Discrete-Event Simulation*", Simulation Practice and Theory Volume 5, Number 2, Pages: 167-184 February 1997.

[2] "http://www.niwotridge.com/Resources/DomainLinks/Ariane5Failure.htm"
J. L. Lions et al, *ARIANE 5 Flight 501 Failure*, Report by the Inquiry Board, July 1996.

[3] Abdel-Moadty M Fayek, "*Introduction to Combined Discrete-Continuous Simulation Using SIMSCRIPT II.5*", CACI International Inc 2000-2005.

[4] J. Frederick Klingener, "*Programming Combined Discrete-Continuous Simulation Models for Performance*", Winter Simulation Conference Proceedings of the 28th conference on Winter simulation, Coronado, California, United States, Pages: 833 – 839, 1996.

[5] Jerry Banks, John S. Carson, "*Introduction to Discrete-Event Simulation*", Winter Simulation Conference Proceedings of the 18th conference on Winter simulation, Washington, D.C., United States, Pages: 17 – 23, 1986.

[6] Herbert Praehofer and Alice Schoeppl, "*A Continuous and Combined Simulation Platform in Java and Its application in Building Paper Mill Training Simulators*", Department of System Theory and Information Engineering Johannes Kepler.

[7] P. Hollands, "*The Use of Radar Environment Simulation for Operator Training and System Test and Evaluation*", Electronic Warfare Systems, IEE Colloquium, Pages: 6/1 - 6/5, 14 Jan 1991.

[8] R.J. Packer, "*Computer Modeling of Advanced radar Techniques*", Electronic Warfare Systems, IEE Colloquium, Pages: 7/1 - 7/6, 14 Jan 1991.

[9] Kiyoshi Sueuchi, Keigo Kotoh, and Takeshi Yajima, "*Computer Simulation of Marine Search and Rescue Operations Using Newly Developed radar Beacons*", Annual Simulation Symposium Proceedings of the 13th annual symposium on Simulation, Pages: 207 – 226, 1980.

[10] Ronald D.Painter, "*Object-oriented Military Simulation Development and Application*", Winter Simulation Conference Proceedings of the 27th conference on Winter simulation Arlington, Virginia, United States, Pages: 1185 – 1188, 1995.

[11] Jane Harmon, Jay Landreth, Donald Lausch, Padman Nagenthiram, PhD., and Henry Ramirez, "*A Development Methodology Applied to a radar System Simulation*", Winter Simulation Conference Proceedings of the 15th conference on Winter Simulation - Volume 2, Arlington, Virginia, United States, Pages: 409 – 418.

[12] "http://www.sydac.com.au/", SYDAC Pty Ltd, South Australia.

[13] "http://www.roke.co.ul", Real Time Radar Simulator, Roke Manor Research Limited 2002.

[14] "http://www.raytheon.com", Raytheon's FIRSTplus ATC Simulation System.

[15] "http://www.telephonics.com ", Telephonics Corporation 2002, Farmingdale, New York 11735.

[16] "http://users.skynet.be/atcsim/", ATC Interactive, Victor DAY.

[17] "http://www.artmacs.com/", Canadian Data Software, Markham, Ontario, Canada.

[18] "http://www.lightmaster.co.uk/", Radar Training simulator, LightMaster Software.

[19] "Secondary Surveillance Radar", Michael C. Stevens, Artech House, 1988.

[20] "http://www.itnu.de/radargrundlagen/top-en.html", Dipl. Ing. (FH) Christian Wolff.

[21] ASR-23SS Series L-Band Solid-State Primary Surveillance Radar, Raytheon company 2004.

[22] "http://www.ales.cz/pdf/radar_systems_morad_l_2d_psr.pdf", MORAD-L, ALES a.s, Slovak Republic.

[23] "http://atcsl.tripod.com/radar_theory.htm", Lanka N.Bogoda, Electronic Engineer, Bandaranaike International Airport, August 2004.

[24] "*A Component-based approach to modeling and simulating mixed-signals and hybrid systems*", JIE LIU and Edward A.LEE, ACM Transactions, Vol. 12, No.4, Pages: 343-368, October 2002.

[25] Philippe B. Kruchten. "*The 4+1 view model of architecture*". IEEE Software, Pages: 42-50, November 1995.

[26] Thomas J. Cheatham, Lee Mellinger, "Testing Object-Oriented Software Systems", Annual Computer Science Conference Proceedings of the 1990 ACM annual conference on Cooperation, Washington, D.C., United States, Pages: 161 – 165, 1990

[27] Y.Lablche, P.Thevenod-Fosse, H. Waeselynck, "*Testing Levels for Object-Oriented Software*", International Conference on Software Engineering Proceedings of the 22nd international conference on Software engineering Limerick, Ireland Pages: 136 – 145, 2000.

[28] D. Kung, J. Gao, P. Hsia, J Lin and Y. Toyoshima, "Class Firewall, test order, and regression testing of object-oriented programs," Journal of Object-Oriented Programming, vol. 8(2), pp. 51-65, 1995.

[29] Gwendolyn H. Walton, Robert M. Patton, Douglas J. Parson, "*Usage Testing of Military Simulation Systems*", Winter Simulation Conference, Proceedings of the 33rd conference on Winter simulation, pp. 771 – 779, 2001.

[30] Robert G. Sargent, "*Validation and verification of simulation models*", Winter Simulation Conference, Proceedings of the 33rd conference on Winter simulation, pp. 39 – 48, 1999.

Appendix – Input Commands of RSS

This appendix describes each kind of command that may be used in a RSS input script. Optional fields in a command are parenthesized, e.g., {transponder status = <transponder status>;}. Names of defined data dictionary items are enclosed within angle brackets, e.g., <transponder status>. These data dictionary items are defined in Appendix B.

Create new aircraft

Format:

<scheduled time> create new aircraft <unique aircraft identifier> [
latitude = <latitude position> degrees;
longitude = < longitude position> degrees;
bearing = <bearing> degrees;
ground speed = <ground speed> nm;
altitude = <altitude>;
{transponder status = <transponder status>; }
{mode_A SSR code = <Mode A SSR code>; }
{mode_C altitude status = <Mode C altitude status>;}
]

Example:

```
00:00:10 create new aircraft bc101 [
  latitude = 23 degrees;
  longitude = 12 degrees;
  bearing = 225 degrees;
  ground speed = 200 nm;
```

```
  altitude = 20000 ft;
  transponder status = off;
  mode_A SSR code = 70;
  mode_C altitude status = on;
]
```

Notes:

Create obstruction

Format:

create obstruction [
width = <width> nm;
height = < height> ft;
rotation = <rotation> degrees;
latitude = <latitude position> degrees;
longitude = < longitude position> degrees;
]

Example:

```
create obstruction [
  width = 100 nm;
  height = 10000 ft;
  rotation = 10 degrees;
  latitude = 23 degrees;
  longitude = 12 degrees;
]
```

Notes:

Create radar source

Format:

<scheduled time> create new radar source <unique radar source identifier> [
latitude = <latitude position> degrees;
longitude = < longitude position> degrees;
elevation = <elevation> ft;
maximum PSR elevation = <maximum PSR elevation> degrees;
maximum SSR elevation = <maximum SSR elevation> degrees;
maximum PSR slant range = <maximum PSR slant range> nm;
maximum SSR slant range = <maximum SSR slant range> nm;
PSR beam width = <PSR beam width> degrees;
SSR beam width = <SSR beam width> degrees;
PSR beam positions = <PSR beam positions>;
SSR beam positions = <SSR beam positions>;
PSR rotations per minute = <PSR rotations per minute>;
SSR rotations per minute = <SSR rotations per minute>;
]

Example:

```
00:01:10 create new radar source kcx [
  latitude = 20 degrees;
  longitude = 10 degrees;
  elevation = 0 ft;
  maximum PSR elevation = 20 degrees;
  maximum SSR elevation = 25 degrees;
  maximum PSR slant range = 100 nm;
```

```
  maximum SSR slant range = 200 nm;
  PSR beam width = 3 degrees;
  SSR beam width = 4 degrees;
  PSR beam positions = 3000;
  SSR beam positions = 4200;
  PSR rotations per minute = 10;
  SSR rotations per minute = 10;
]
```

Notes:

Create station pressure

Format:

<scheduled time> create station pressure aircraft <unique station name> observation

[

latitude = <latitude position> degrees;

longitude = < longitude position> degrees;

elevation = <elevation> ft;

pressure = <pressure>;

]

Example:

```
00:01:43 create station pressure qt17 observation [
  latitude = 5 degrees;
  longitude = 10 degrees;
  elevation = 1000 ft;
  pressure =  29.92;
]
```

Notes:

Modify aircraft Mode A SSR code

Format:

<scheduled time> <aircraft identifier> mode_A SSR code <Mode_A SSR Code>;

Example:

```
00:12:57 bc727 mode_A SSR code 7700;
```

Notes:

Modify aircraft Mode C Altitude Status

Format:

<scheduled time> <aircraft identifier> mode_C altitude status <Mode_C Altitude Status>;

Example:

```
00:22:37 bc727 mode_C altitude status on;
```

Notes:

Modify aircraft altitude

Format:

<scheduled time> <aircraft identifier> altitude <altitude> <altitude change rate feet/min>;

Example:

```
10:48:43 bc727 altitude 20000 500;
```

Notes:

Modify aircraft bearing

Format:

<scheduled time> <aircraft identifier> bearing <bearing> <+/-bearing change rate degrees/min>;

Example:

```
22:02:00 bc727 bearing 120 -10;
```

Notes:

Modify aircraft ground speed

Format:

<scheduled time> <aircraft identifier> ground speed <ground speed> <+/- acceleration nm/h/min>;

Example:

```
07:52:23 bc727 ground speed 500 20;
```

Notes:

Modify aircraft transponder status

Format:

<scheduled time> <aircraft identifier> transponder status <transponder status>;

Example:

```
03:32:57 bc727 transponder status off;
```

Notes:

Delete aircraft

Format:

<scheduled time> delete aircraft <aircraft identifier>;

Example:

```
05:00:00 delete aircraft bc727;
```

Notes:

Modify radar source status

Format:

<scheduled time> <radar source identifier> <radar source status>;

Example:

```
04:17:53 ktx PSR_only;
```

Notes:

Terminate simulation

Format:

<scheduled time> terminate simulation;

Example:

```
01:22:07 terminate simulation;
```