
# ADVANCED FRAMEWORK FOR SIMULATION, INTEGRATION AND MODELING (AFSIM)

## Version 2.0

## OVERVIEW AND TECHNICAL REFERENCE

**James Zeh, AFRL/RQQD**
**Brian Birkmire, AFRL/RQQD**

**Nicholas A. Chinnici**
**Peter D. Clive**
**Jeffrey Johnson**
**Andrew W. Krisby**
**Jonathon E. Marjamaa**
**Luke B. Miklos**
**Michael J. Moss**
**Stephen P. Yallaly**

**August 2016**

**AIR FORCE RESEARCH LABORATORY**
**AEROSPACE SYSTEMS**
**WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7801**
**AIR FORCE MATERIEL COMMAND**
**UNITED STATES AIR FORCE**

TABLE OF CONTENTS

**Section** **Page**

# List of Figures & Tables

**Figure** **Page**

# 1.0 OVERVIEW

## 1.1 Introduction

AFSIM is a government-approved software simulation framework for use in constructing engagement and mission-level analytic simulations for the Operations Analysis community. The primary use of AFSIM applications is the assessment of new and advanced system concepts, and the determination of concepts of employment for those systems.

The framework provides the ability to model the capabilities of the participants and to control the interaction of the participants as they move through space and time. The resulting simulations can be:

- Constructive/non-interactive (the user invokes the simulation which then runs without further interaction), or interactive (the user or other simulation controls some aspects of the simulation.
- Non-real-time (faster or slower depending on the fidelity of the platform component models), or real-time (constrained by some multiple of a real-time clock).
- Event-stepped (simulations proceed according to processing of relevant events) or time-stepped (simulations proceed according to events occurring in succeeding time steps)

AFSIM is designed to be generally usable "as is," so that not only are top-level modeling concepts defined by the framework, but many concrete implementations are also provided. Examples of standard models delivered with the framework include the following:

- Movement models
- Sensor systems
- Weapon systems and weapon effects
- Communication systems
- Information processing systems (trackers, etc.)
- Decision making systems (command and control, missile guidance, etc.)
- Antenna pattern models
- Atmospheric attenuation models
- Signal propagation models
- Clutter models
- Electronic warfare effect models

AFSIM is also extensible, providing for ease of incorporating models of the above types, as well as completely new capabilities. Because AFSIM is a framework, simulation details and services are provided, freeing the developer to focus specifically on adding desired new functionality through implementation of the framework's abstract interfaces. Additionally, AFSIM's Component-Based Architecture (CBA) provides the ability to cleanly and generically incorporate nearly any new modeling capability into the framework. Once incorporated, the models are then as much a part of the framework as any of the standard models.

AFSIM simulations are typically used to evaluate the performance of military systems in the context of a mission. To be successful, the framework must provide the capability to model the performance of the participants within the environment of the missions. In AFSIM, the individual participant is referred to as a *platform*, which in some simulations is called an entity. Platforms represent things such as aircraft, satellites, missiles, ships, submarines, ground vehicles, structures and life-forms. The platform contains components such as communications, sensors and weapons systems, and information and decision-making systems. These components are used to gather, process, and disseminate information, make command decisions, and carry out the commands. The framework, with its supplied component models, as well other component models that may have been added, provides the capabilities to model the platforms participating in the simulation.

This section provides a general overview of the Advanced Framework for Simulation, Integration and Modeling (AFSIM). Section two provides a description of the standard AFSIM software distribution, including standard AFSIM applications, the AFSIM Integrated Development Environment (IDE), AFSIM IADS scenarios, and Visual Environment for Scenario Preparation and Analysis (VESPA). Specific algorithms and detailed model descriptions are covered in the appendices of this document, which together serve as the AFSIM technical reference guide. This document does not, however, provide detailed descriptions of user inputs, the AFSIM scripting language, or the IDE. These are detailed in the AFSIM Wiki, which serves as the reference manual for AFSIM commands. Similarly, learning to use AFSIM is accomplished either by taking the AFSIM Analyst course or through study of the many examples and demonstrations that accompany the AFSIM release.

## 1.2 Platforms

AFSIM platforms are composed of attributes, information, components and links (see Figure 1). Attributes include the name, type, affiliation and commander of the platform. Other physical attributes include its radar, optical and infrared signatures, defining its susceptibility to being seen by a sensor at a specified aspect angle. Information stored on the platform includes measurements, tracks, tasks, and human perceptions; these form the basis upon which platforms make decisions. Platform components include definitions of movement, sensors, communications, weapons and processing (either to model mental or computing components), which are described individually in the following sections. Finally, internal and external links provide communication via messages.

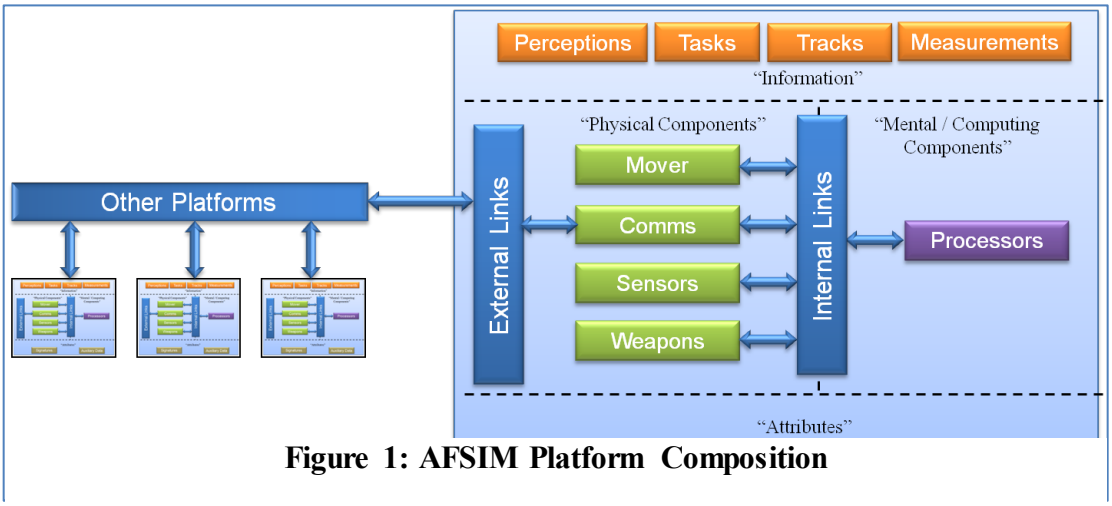


**Figure 1: AFSIM Platform Composition**

Internal links tie individual internal platform components together in a flexible way. External links provide inter-platform communication through communication devices.

## 1.3 Platform Components

Platform components provide the models for the physical and mental/computing operating capabilities of a platform. Many standard component models are provided with the framework, some of which are described in the following paragraphs.

Component models delivered with AFSIM are often provided at various levels of fidelity. For example, an aircraft can be modeled with a simple route mover or with a full pseudo 6-DOF (six degrees of freedom) aerodynamics and control model. Another example is communications models that can either deliver messages perfectly, or that are based on electromagnetics and subject to signal loss. These choices provide the flexibility to easily configure a simulation using lower-fidelity models, or, to provide necessary effects with higher fidelity ones.

AFSIM also allows new component models (sensors, weapons, etc.), as well as completely new component types to be inserted and utilized. Although beyond the scope of this document, describing and teaching this process is a main theme of the AFSIM Software Developer Course.

### 1.3.1 Movers

A mover is an optional component providing the means by which a platform moves through space-time (a platform without a mover is geographically fixed). The simulation affects the movement of platforms and ensures a platform's position is current before involving it in any interactions with other platforms or the simulated environment.

There are many types of standard mover models, providing the capability to represent platforms in every domain, from seabed to space. Some examples of movers delivered with AFSIM include the following:

- WSF_AIR_MOVER – A route mover for air vehicle motion
- WSF_GROUND_MOVER – A route mover for a terrain-following ground vehicle
- WSF_ROAD_MOVER – A ground mover that moves on a road network and is able to traverse a shortest path along it
- WSF_SURFACE_MOVER – A route mover for a surface ship
- WSF_SUBSURFACE_MOVER – A route mover for a submersible vehicle
- WSF_NORAD_SPACE_MOVER – A mover for a platform in orbit about the Earth
- WSF_GUIDED_MOVER – A mover that is capable of representing a guided glide bomb or a single or multistate guided missile
- WSF_TSPI_MOVER – A mover that updates position based on Time Space Position Information (TSPI) data read from a text file
- WSF_FIRES_MOVER – A mover for an indirect fire (rocket, artillery, mortar) round
- WSF_P6DOF_MOVER – A high-fidelity, pseudo-6DOF mover, providing angular and translational kinematics

### 1.3.2 Sensors

Sensor systems are used to sense the environment around platforms. There are many types of standard sensor models, some of which are:

- WSF_ACOUSTIC_SENSOR - Baseline acoustic sensor model
- WSF_AMBER_SENSOR - Provides an interface to the TMAP AMBER radar models
- WSF_EOIR_SENSOR - Electro-Optical/Infrared (EOIR) sensor model
- WSF_ESM_SENSOR - Baseline passive RF detection sensor
- WSF_GEOMETRIC_SENSOR - Baseline sensor based purely on geometry
- WSF_IRST_SENSOR - Baseline infrared search-and-track sensor
- WSF_OPTICAL_SENSOR - Baseline optical sensor model
- WSF_OTH_RADAR_SENSOR - Baseline Over-The-Horizon Backscatter (OTH-B) sky wave radar model
- WSF_RADAR_SENSOR - Baseline radar model
- WSF_SAR_SENSOR - Baseline synthetic aperture radar (SAR) model
- WSF_SOSM_SENSOR - Interface to the Spectral Optical (IR) Sensing Model (SOSM)
- WSF_SURFACE_WAVE_RADAR_SENSOR - Over-the-horizon radar surface wave sensor model

Many sensors, such as the RADAR sensor models, implement modeled electromagnetic interactions and are subject to jamming.

### 1.3.3 Communications

Communications systems are used to transmit information from one platform to another. In addition to model choice, the framework also provides a rich set of networking options, detailed in the next section. AFSIM offers three system models:

- WSF_COMM_TRANSCEIVER – Implements perfect or wired communications
- WSF_RADIO_TRANSCEIVER – Implements radio frequency communications
- WSF_JTIDS_TRANSCEIVER – Implements JTIDS/Link 16 communications

WSF_RADIO_TRANSCEIVER and WSF_JTIDS_TRANSCEIVER both implement modeled electromagnetic interactions and are subject to jamming.

### 1.3.4 Weapons

Weapons systems are used to either lethally or non-lethally attack another platform or platform component. Weapons can utilize launch computers to make firing decisions, and they employ configurable weapon effects. AFSIM weapons can be either *explicit* or *implicit*. Explicit weapons create new (weapon) platforms representing missiles, bombs, etc., when fired. Implicit weapons do not create weapon platforms and are used to model non-kinetic weapons such as high-energy lasers and jammers, as well as kinetic weapons with predictable trajectories, such as artillery.

The weapon components delivered with AFSIM are the following:

- WSF_EXPLICIT_WEAPON represents a weapon that is modeled as an independent platform when it is fired.
- WSF_IMPLICIT_WEAPON represents a weapon that does not require an independent flyout.
- WSF_RF_JAMMER is a radio frequency jammer that can be used to disrupt radar, ESM and radio communication systems.
- WSF_LASER_WEAPON represents a High-Energy Laser (HEL) weapon with an independently configurable laser fluence model.
- WSF_CUED_LASER_WEAPON is an implementation of WSF_LASER_WEAPON that is cued by a separate beam director sensor.

Note that there is only one explicit weapon model. This is because the wide range of possibilities for missiles, bombs, etc., is achieved by configuring weapon platform types with their own unique platform components. For example, a missile can be specified as a weapon platform type configured with a guided mover and guidance computer (processor), and it is the mover and guidance computer that are configured to model the unique characteristics of the missile.

### 1.3.5 Processors

Processors are used to implement the mental or computing processing capabilities of a platform. There are many processors available in the standard framework, among the most common being:

- WSF_SCRIPT_PROCESSOR – A general purpose processor for executing scripts, written with the AFSIM scripting language, to implement custom behaviors and custom processing
- WSF_TASK_PROCESSOR – A scriptable, finite-state machine for making command and control decisions about the tracks known to a platform.
- WSF_RIPR_PROCESSOR – A more advanced scriptable processor for making command and control decisions
- WSF_GUIDANCE_COMPUTER – Implements a guidance computer for missiles
- WSF_GROUND_TARGET_FUSE and WSF_AIR_TARGET_FUSE – Implements a fusing mechanism for a weapon platform
- WSF_IMAGE_PROCESSOR – Simulates analysis of 'images' produced by imaging sensors
- WSF_MESSAGE_PROCESSOR – A scriptable router and interpreter of messages
- WSF_TRACK_PROCESSOR – Accepts tracks from on-board and off-board sources and feeds them to the track manager. Also sends periodic updates of local tracks to other platforms

Depending on its implementation, a processor may be invoked periodically or as the result of some event such as the reception of a message or the expiration of a time interval.

### 1.3.6 Other Components

AFSIM platforms can also utilize other component types, such as the following:

- Command Chain – Provides the platform's command and control reporting structure
- Fuel – Provides fuel consumption and refuel capability
- Intersection Mesh – Provides a representation of the platform's detailed geometry, against which ray-tracing calculations can be made
- Navigation Errors – Models the platform's computed location
- Visual Part – Provides a component that can be used for visualization purposes and to compute geometry

If a new component type is needed that is not provided with the standard AFSIM distribution, it can be integrated using the AFSIM's Component Based Architecture (CBA, newly introduced in version 2.0). AFSIM CBA provides an abstract platform component interface that is extended to easily add new and existing base component types, as well as component "extensions" that add additional functionality for existing sensor, processor, and weapon components. Using CBA, integrating new and existing capabilities on platforms is not only generic and simple, but using and understanding them is often much easier as well.

## 1.4 Additional Platform Capabilities

### 1.4.1 Electromagnetic Interactions with Transmitters and Receivers

Representations of transmitters and receivers are used by platform components (sensors, communications, and certain types of weapons) to simulate the transmission and reception of electromagnetic (EM) radiation. The framework accomplishes this simulation through ensuring a consistent set of EM interactions between transmitters and receivers. Utilizing this software architecture, AFSIM naturally allows for such effects as jamming and passive detection.

### 1.4.2 Antenna Patterns

Antenna patterns are attached to transmitters and receivers. A simple antenna pattern is a two-dimensional table that provides gain as a function of azimuth and elevation off the pointing angle of the antenna. The framework allows patterns to be defined using two-dimensional tables or using one of several algorithmic patterns. There are also several other optional sensor models that provide additional pattern models. A polarization and frequency-dependent antenna pattern may also be defined.

### 1.4.3 Attenuation, Propagation, Fluence, and Clutter Models

Attenuation models are available to limit (attenuate) propagated EM signal. Among the attenuation model choices available in the framework are several standard radar-specific options as well as a general model for optical attenuation.

Propagation models in AFSIM are designed to model specific effects at radar wavelengths; options for multipath, ground wave propagation, and knife edge effects are available.

Fluence models simulate high energy laser propagation. AFSIM provides a core fluence model as well as an interface to the industry-standard HELCoMES fluence model.

AFSIM also provides the ability to employ a low-angle clutter model to simulating radar backscatter, and the resulting noise, from terrain. These models utilize the full set of clutter backscatter coefficients from MIT/Lincoln Labs studies.

### 1.4.4 Electronic Warfare Effects

Transmitters associated with systems such as RF jammers (e.g., WSF_RF_JAMMER) may define the abilities of the transmitter to electronically attack receivers, thus implementing Electronic Attack (EA) techniques. Similarly, receivers associated with sensors or communications systems (e.g., WSF_RADAR_SENSOR or WSF_RADIO_TRANSCEIVER) may define the ability of the receiver to mitigate the effects of an electronic attack, thus implementing Electronic Protect (EP) techniques. The list of available EA and EP techniques, and their effects, is fully described in Appendix D, "AFSIM EW Architecture."

### 1.4.5 Communications Networks

AFSIM provides the capability to model the network associated with the passing of messages by communications devices. Multiple communications devices act as nodes on a network, and networks can have gateways to other networks. Messages are intelligently routed from source to destination. One can specify a link-layer protocol for realistic transmission times. Message queuing and filtering are also supported.

### 1.4.6 Link-16 Tadil-J

AFSIM is delivered with the capability to send and receive Link-16 Tadil-J messages over communications devices. Received messages are processed with user-defined AFSIM scripts. One may also send and receive these messages over distributed simulation interfaces, so that AFSIM platforms can participate as nodes in a virtual distributed command and control network.

### 1.4.7 Tracking and Filtering

The track is the fundamental object that represents the information known about another object such as a platform or a transmitter. That information may describe the physical state of an object (location, velocity, etc.), or it may describe other features (side, type, etc.). The object may or may not be real, and the information generally contains errors. Thus, a track represents the imperfect knowledge of what a platform perceives about another object. It is the primary input to most of the decision-making process, and because it can represent flawed knowledge, poor decisions can be made.

The amount of information in a track depends on its source and filtering or fusion processes that may have been applied. A sensor may provide range and bearing, or it may only produce range bearing and elevation. It may also provide affiliation and type (e.g., IFF), measurement quality, etc.

Overwhelmingly, tracks represent the products of sensor measurements. Multiple measurements must be associated, or "correlated" with each other to create and update a track. This correlation process presents the opportunity to mistakenly group measurements with incorrect objects or with no object at all. In the latter case, false targets are produced. AFSIM offers several options for correlation, giving the analyst the option to either explore or ignore such effects.

A primary role of tracking is to provide as accurate an estimate as possible to truth. It is the job of filtering to provide these estimates, especially for the kinematic data. For this purpose, AFSIM includes a standard set of filters, providing location and velocity state estimates. The framework is also flexible enough to allow integration of nearly any kind of filter.

The other very important goal of tracking is to intelligently combine data from measurements and tracks to produce as complete an operational picture as possible. Often, new data will be computed or inferred in this process called fusion. AFSIM is delivered with a standard fusion engine, and the framework allows substitution with other fusion algorithms as desired.

### 1.4.8 Tasks

The AFSIM task management service provides the capability to send and receive task assignments, perform the tasks, and provide the status information for dealing with those assignments. Tasks are usually associated with a track; for example, initiate a sensor tracking request cued to a track location, fire a weapon at the given track location, etc.

### 1.4.9 Behavior Modeling

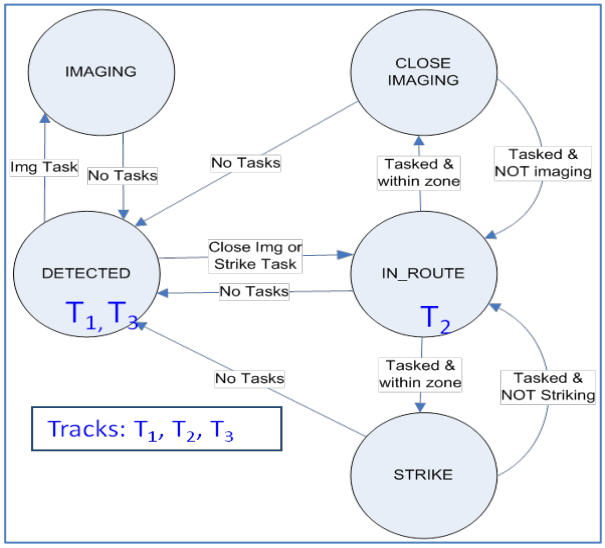


**Figure 2: Graphical representation of an example task processor state machine.**

The execution of tasks is performed with the AFSIM scripting language, and this is the most common method by which platform behaviors are implemented. Two options provided by the framework for task execution are the WSF_TASK_PROCESSOR and the WSF_QUANTUM_TASK_PROCESSOR. The WSF_TASK_PROCESSOR utilizes a finite state machine, for which tracks move between states, and scripts implement behaviors based on the current state (see Figure 2, in which the circles represent states and arrows represent transition rules). The WSF_QUANTUM_TASK_PROCESSOR utilizes a behavior tree (Figure 3), an artificial intelligence technology in which behaviors are created and arranged as nodes in a graph or "tree." The behavior nodes can be arranged together in interesting and interrelated ways, providing more flexibility in how behaviors are selected and executed. The artificial intelligence architecture and behavior trees are described further in Appendix D.

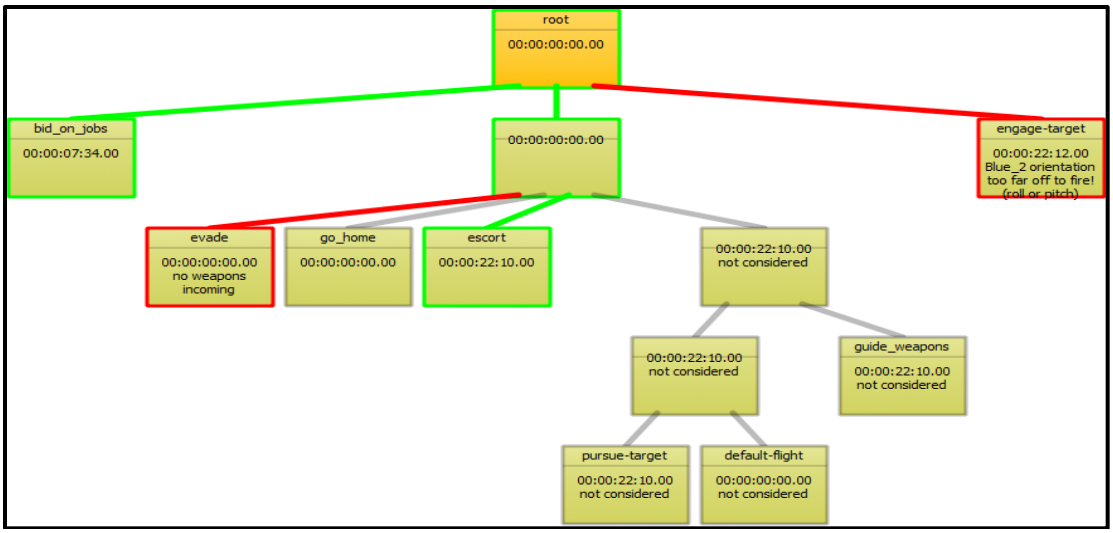


**Figure 3: An Example RIPR Behavior Tree**

## 1.5 Simulation Services

### 1.5.1 Scripting Language

The AFSIM scripting language provides a mechanism for the user to execute a complex set of instructions based upon events that occur in the simulation. The language is similar to C# and Java and should be familiar to anyone with basic programming skills. It is block-structured and contains familiar declaration, assignment, and flow control statements that allow the user to examine and manipulate the simulation environment. Some uses of the scripting language include the following:

- Implement tactics and doctrine
- Data collection
- Dynamic configuration of objects based on mission options
- Simulation control

### 1.5.2 Terrain and Line-of-Sight Management

Terrain is used by the framework to perform such actions as determining if objects involved in interactions are obscured by terrain, determining if an airborne object crashes into the ground, or to constrain a ground vehicle to the ground. Acceptable forms of terrain data include National Geospatial Agency (NGA) Digital Terrain Elevation Data (DTED) and Arc-Info Float Grid format.

### 1.5.3 Event Logging and Standard Output

The framework implements an "observer" capability that allows simulation components to be notified of the occurrence of certain events (e.g.: turning a sensor on, sensor detection attempts, firing a weapon, sending a message assigning a task, etc.). This capability is used to implement event logging without having to modify all sections of the code. Several forms of standard output are provided. Additionally, AFSIM provides the capability for the user to write scripts to perform any desired functions such as writing a custom output file or accumulating statistics.

### 1.5.4 Extensions and Plug-Ins

Extensions and plug-ins are the primary mechanisms by which the framework is extended to integrate new platform component models, new and extended platform capabilities, and new and extended simulation services. The plug-in capability is a form of extension that allows one to add capabilities without re-compiling the core AFSIM code. Use of plug-ins allows for easier distribution of extended capabilities, and they provide the ability to choose which extended capabilities to use for a given analysis.

### 1.5.5 Distributed Simulation Interfaces

The observer capability is also utilized to implement distributed simulation interfaces. AFSIM provides three such standard interfaces:

- DIS (Distributed Interactive Simulation) interface, which allows an AFSIM simulation to participate as one application in a DIS exercise.
- High Level Architecture (HLA) interface, which allows an AFSIM simulation to participate as one HLA federate in a HLA exercise.
- XIO (eXternal Input/Output) interface, providing a multi-machine capability, wherein two or more AFSIM simulations interact without the limitations of DIS and HLA. XIO can also be used to interface AFSIM simulations with GUIs and other user input devices, providing operator-in-the-loop capabilities.

### 1.5.6 Monte Carlo Iteration and Design of Experiments

AFSIM provides the capability to perform a set of simulation runs at a time, where each run starts with a different random number seed, and variability is introduced among the runs due to the differing set of random numbers produced. This is often called *Monte Carlo iteration*. There are many options within AFSIM to introduce this desired variability among runs, enabling its use in design of experiments (DOE) studies.

## 1.6 Limitations and Assumptions

There are very few constraints on the bounds of the mission. The number of platforms is limited only by the amount of memory available and the amount of time one is willing to wait. The mission may cover the whole world from underwater to space. Mission duration can be very long: months or years.

There are no inherent limitations imposed by the framework on platforms or their components with regard to their performance. Any platform can detect, communicate with, shoot or control any other platform. Any limitations are due either to the how the model is configured (user input) or limitations of a platform component model. Such limitations can be overcome through

- Changing the parameters of the existing platform component model to reflect the desired fidelity.
- Choosing a different platform component model with the desired fidelity.
- Acquiring a different platform component model with the desired fidelity.

For example, unless otherwise indicated, a sensor will attempt to detect all other platforms in the simulation. The user can make the assumption that it can distinguish friends from enemies and prevent detection attempts against friendly platforms, thus saving a lot of computation time. Another example is that a user may choose WSF_GEOMETRIC_SENSOR to model a radar system very simply (e.g. if the target is within 50 miles it can be seen). The user has assumed that this is a viable assumption, but if proven otherwise, the user can choose WSF_RADAR_SENSOR for more fidelity. The point is that the user is in charge of most limitations of this kind.

In order to complete a mission, platforms must have some sort of information on which to act. The truthfulness and accuracy of the information often governs the success of the mission. Assume a commander receives information that a target is located at a specific location and commands an asset to destroy the target. If the location had significant errors, or if the target did not really exist (it was a false report or the target had already been destroyed and the commander had not been informed), time and resources would have been wasted chasing a target that wasn't there, thus potentially preventing the destruction of a real target. The framework imposes no inherent limitation on the viability of the information in a track. It can contain errors and does not have to correspond to a real platform.

# 2.0 AFSIM SOFTWARE DISTRIBUTION

## 2.1 AFSIM Applications

### 2.1.1 WSF Exec

The baseline AFSIM simulation application is called the World Simulation Framework Executive (Wsf Exec). Wsf Exec reads a user-defined input file, executes the simulation, and outputs any user-defined data files. As Wsf Exec incorporates all standard AFSIM features, it is used to execute all simulation demos and scenarios included with AFSIM releases, including the AFSIM IADS scenarios. Wsf Exec also most often provides the simulation engine used by the AFSIM IDE to execute simulation scenarios within an information-rich, graphical context.

*Note: Wsf Exec is likely to undergo a name change, to become "AFSIM Mission."*

### 2.1.2 Sensor Plot

Sensor Plot is an AFSIM application that is used to evaluate sensor characteristics and interactions with user-specified geometries. It has the ability to create files that produce plots of

- Sensor vertical coverage
- Sensor horizontal coverage
- Sensor spherical coverage
- Defended area of a collection of sensors
- Antenna patterns

### 2.1.3 Weapon Tools

Weapon tools is an AFSIM application that repeatedly fires a predefined WSF_EXPLICIT_WEAPON over widely varied engagement conditions for the purpose of creating the various launch computers required for making acceptable weapon firing decisions. It currently

allows for creation/generation of launch computers for air/air, air/ground, and ballistic missile (ground/ground).

## 2.2 AFSIM IDE

The AFSIM Integrated Development Environment (IDE) supports the analyst in defining and integrating AFSIM platforms and platform components. The AFSIM IDE patterns itself on IDEs created for use with software development. With software IDEs, a single application is used to edit files, compile, link, and run the software executable, and view output results or error messages. Likewise, the AFSIM IDE permits the analyst to edit input files, execute an AFSIM application, and visualize the output results and any error messages. The iterative process that allows the analyst to receive immediate feedback as platform and component models are defined and scenarios created is illustrated in Figure 4.

*Note: AFSIM IDE is likely to undergo a name change, to become "AFSIM Wizard."*

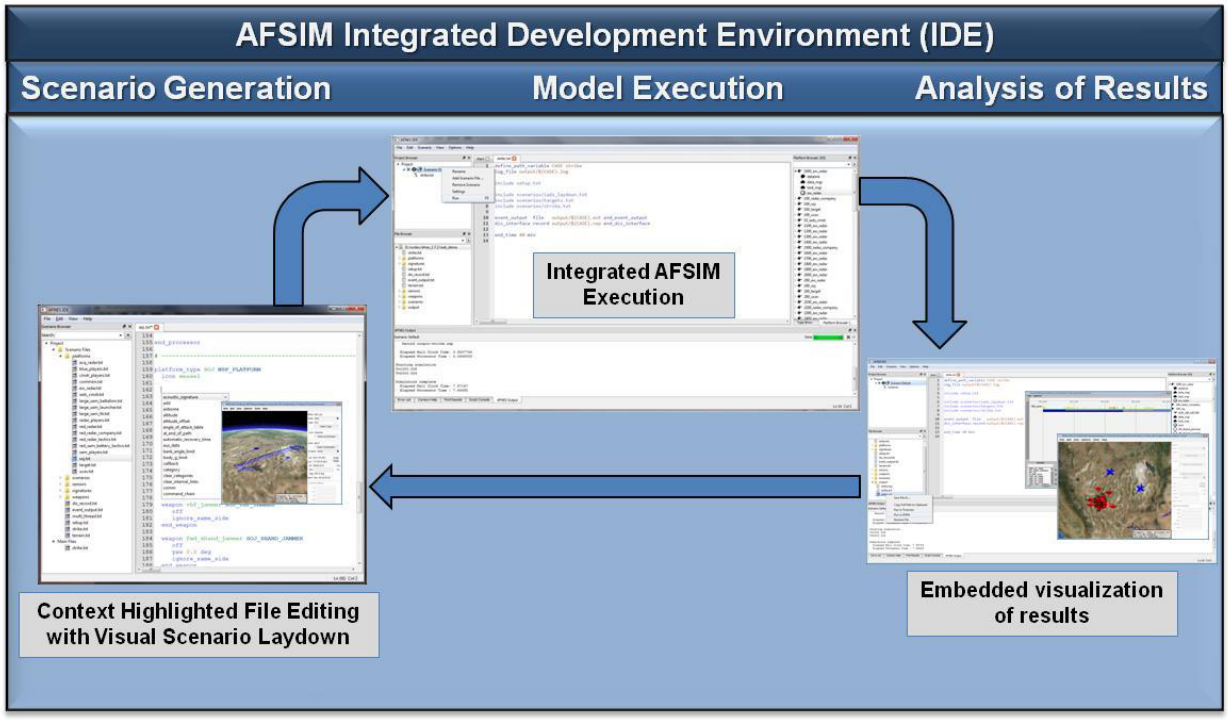


**Figure 4: AFSIM Constructive Analysis Process Using the AFSIM IDE**

Current capabilities of the AFSIM IDE to support input file creation include file syntax highlighting, auto-completion, context-sensitive command documentation, a variety of scenario browsers, and a script debugger. Syntax highlighting makes reading and understanding the content easier for the analyst. Unknown keywords or commands are underlined in red for easy discovery. Examples of unknown keywords or commands include misspelling of keywords or using keywords out of scope. The auto-completion feature provides a list of suggestions for the analyst to choose from, based on the context. The analyst can select one of the suggestions, and the command will be completed without having to manually type the command. Context-sensitive command documentation allows the analyst to bring up documentation associated with a command to

illustrate the scope and use of the command. Other AFSIM IDE capabilities are available to assist the analyst in defining platform and component models and scenarios.

With the IDE's built-in Demo Browser, analysts can look through over thirty example scenarios. These scenarios demonstrate the vast capabilities of AFSIM broken down into smaller, focused pieces. The demos serve as a great starting point for creating new scenarios as well as reference for adding to existing scenarios.

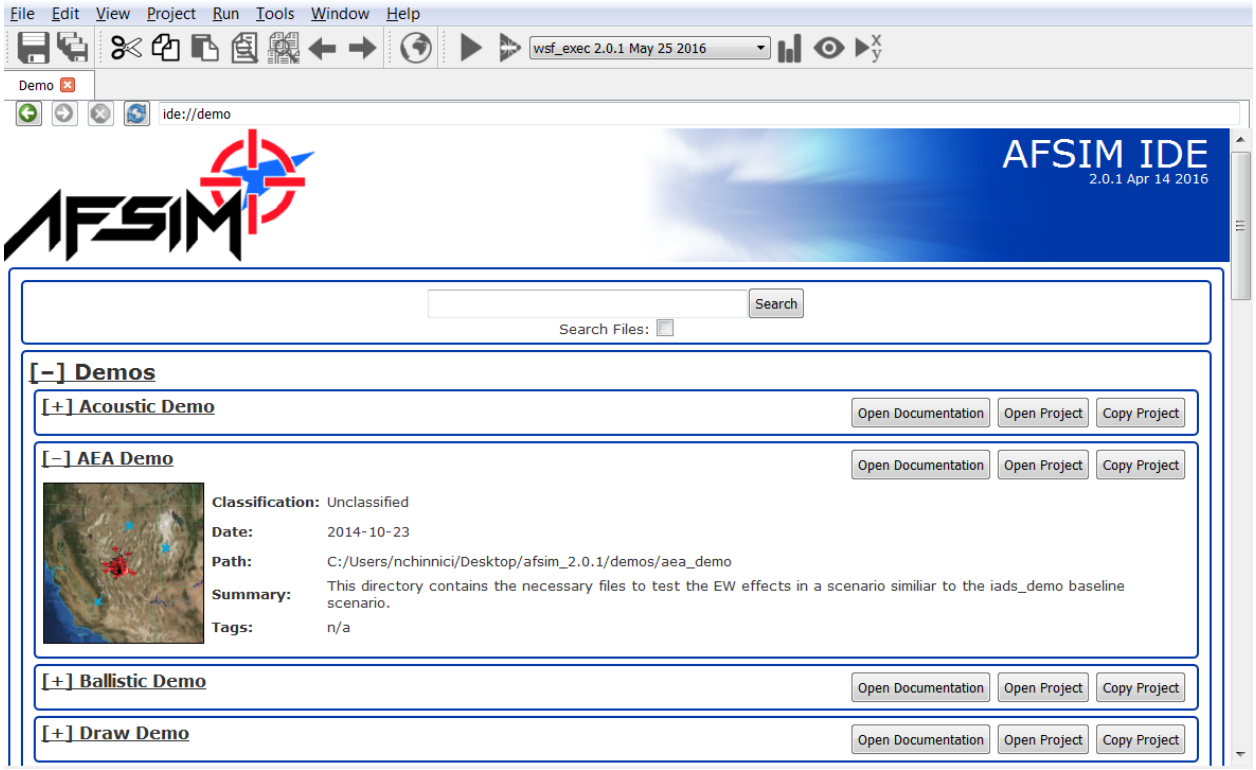


**Figure 5: The AFSIM IDE Demo Browser**

The AFSIM IDE can execute any AFSIM-based application using the input files defined by the analyst. Any screen output from the application is displayed in an IDE output window along with any error messages.

Current capabilities of the IDE to view simulation results include the ability to run the VESPA application from the IDE using the AFSIM replay file created during the simulation run.

A general plug-in capability is also part of the AFSIM IDE. This supports the development of features that are needed by a limited user set. This feature includes the plug-in interface to the application, and user-interface to manage plug-ins within the applications.

## 2.3 VESPA

The Visual Environment for Scenario Preparation and Analysis (VESPA) software application enables creation of scenario initial condition files compatible with any AFSIM-based application. In addition, VESPA can be used to visualize object positional time histories and other

event information generated as output from any AFSIM-based application. This allows the analyst to quickly understand and analyze the output from the simulation. Since VESPA is a "DIS-listener" visualization tool, it may also be used to display real-time entity interactions from any real-time simulation that publishes DIS data.

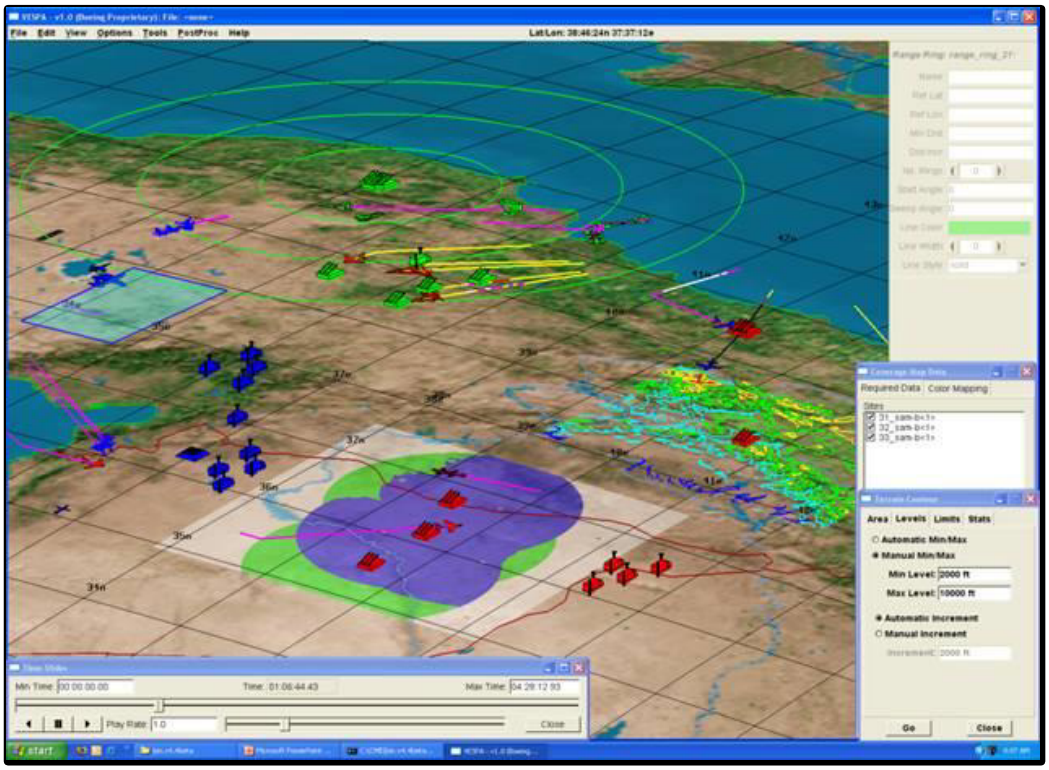

**Figure 6: VESPA GUI**

VESPA includes a graphical user interface (GUI) that includes a drawing area with a geospatial map and a data input area, as shown in Figure 5.

Using VESPA, the analyst can place icons representing objects at specific latitude and longitude locations on a geospatial map. Initial conditions can then be assigned for each selected object. For example, the initial conditions of an aircraft could be its speed, heading and altitude. Visual features associated with objects, called attachments, can also be created. Examples include routes, range rings and zones.

VESPA can be used to display object positional histories and events using an AFSIM replay file generated during an AFSIM simulation run. The AFSIM replay file is a binary file containing the DIS output from the AFSIM simulation. In addition, plots can be generated for selected events that occurred during the simulation.

## 2.4 Standard IADS Scenarios

The AFSIM classified distribution includes several standard IADS (Integrated Air Defense System) based scenarios. Algorithms and software from the Air Force mission level model were incorporated into the AFSIM, including sensor modeling algorithms, the missile flyout code, and the command chain logic in standard scenarios. Both the scenario data base (SDB) and type data base (TDB) files for these scenarios have been translated into AFSIM input format, and all Air

Force mission level model updates are then performance verified with the AFSIM IADS. This includes comparisons of sensor vertical coverage diagram (VCD) plots for all sensor types in these scenarios, as well as missile flyout comparisons for all surface-to-air missile (SAM) types. The result is a continuous verification and validation (V&V) process to standard scenarios. Additional information on the pedigree of the AFSIM IADS model can be requested by contacting the AFSIM model manager.

## 2.5 Target Machine Requirements

AFSIM applications are currently distributed as 32-bit and 64-bit Linux and Windows executables. As they can be run from the command line, there are few restrictions on the target machine's configuration, other than the operating system. However, because the IDE and VESPA are graphical applications, there is the added restriction that the target machine should possess a relatively modern graphics chip or card. Although compatibility is not guaranteed, VESPA and IDE will generally function well with graphics chips/cards manufactured on or after 2005. Four or more GB of memory is recommended.

## 2.6 Distribution

Distribution of AFSIM is coordinated through AFRL/RQQD (Aerospace Systems Directorate, POC Brian Birkmire). An Information Transfer Agreement (ITA) must be completed to receive either the unclassified or classified distribution, and an appropriate DD254 must be presented by government contractors to acquire the classified distribution.

# Appendix A- AFSIM Communication, Sensor and Jamming Equations

## A.1 Overview

The purpose of this document is to describe the equations and algorithms used in the interactions between objects in AFSIM. This includes:

- Sensor interactions
- Communication interactions
- Disruption (jamming) interactions

## A.2 Common Radio Frequency Equations

AFSIM utilizes a common set of classes to encapsulate the components involved in radio frequency (RF) interactions (in reality, some features of these classes are also used for non-RF interactions, but that is not important here). The first section of the document will deal with the basics of signal transmission and reception. Subsequent sections of the document will deal with specific uses (radar, SAR, ESM, jamming, communications).

Ignoring the details of the processing of the received signal, RF interactions fall into two classes:

- Direct or one-way, i.e., an emitted signal going directly to a receiver
- Indirect or two-way: i.e., an emitted signal reflected from an object and then received

The computation of the received signal power can be broken into distinct steps:

- Emission from the transmitting antenna
- Propagation to the target or the receiver
- For indirect or two-way interactions
- Reflection from the target
- Propagation from the target to the receiver
- Reception by the receiving antenna

## A.3 Calculation of direct transmitted power

$$P_x = P_{peak} \times DC \times \frac{G_x}{L_x} \quad \text{(RF.1)}$$

## Table A-1: Transmitted Power Variables

| Symbol | Source | Description |
|---|---|---|
| $G_x$ | transmitter antenna_pattern | The gain of the transmitting antenna in the direction of the target object (receiver or platform). This includes any electronic beam steering losses (Equation RF.6). |
| $L_x$ | transmitter internal_loss | The internal losses in the transmitter between the power source and the antenna. |
| DC | transmitter duty_cycle | The user defined duty-cycle of the transmitter (default: 1.0, if not defined). |
| $P_{peak}$ | transmitter power | The peak power of the transmitter. This should be the power of a single pulse. |
| $P_x$ | Computed | The transmitted power. |

### A.3.1 Propagation of the signal in free space

The propagation of a free space signal from the source (s) to the destination (d) is computed using the following equation. In a one-way interaction, 's' and 'd' are the transmitter and receiver respectively (Equation RF.2b). In a two-way interaction, there are two propagation paths. The first is from the transmitter to the target (Equation RF.2c) and the second is from the target to receiver (Equation RF.2d).

$$D_{sd} = P_s \times \frac{A_{sd}}{4\pi R_{sd}^2} \quad \text{General Form} \quad \text{(RF.2a)}$$

$$D_{xr} = P_x \times \frac{A_{xr}}{4\pi R_{xr}^2} \quad \text{Transmitter - to - receiver} \quad \text{(RF.2b)}$$

$$D_{xt} = P_x \times \frac{A_{xt}}{4\pi R_{xt}^2} \quad \text{Transmitter - to - target} \quad \text{(RF.2c)}$$

$$D_{tr} = P_t \times \frac{A_{tr}}{4\pi R_{tr}^2} \quad \text{Target - to - receiver} \quad \text{(RF.2d)}$$

**Table A-2: Free Space Propagation Variables**

| Symbol | Source | Description |
|---|---|---|
| $A_{sd}$ | transmitter attenuation_model | The fraction of the signal that remains after computing the effects of atmospheric attenuation while propagating the signal from the source (s) to the destination (d). |
| $D_{sd}$ | Computed | The computed free space power density at the destination (d) that originated from the source (s). |
| $P_s$ | Computed | The power emitted from the source (s). This will be either the transmitted power (Equation RF.1) or the reflected power from a target (Equation RF.3). |
| $R_{sd}$ | Computed | The slant range from the source (s) to the destination (d). |

### A.3.2 Reflecting a free space signal

A target that reflects a free space signal effectively creates a new 'transmitting source'. The power of the source is simply the product of the signal density of the incoming signal times the effective area of the reflecting source. The reflector can be a platform (such as when performing a two-way radar interaction) or can be the surface of the earth (when performing clutter calculations). The reflected power can then be propagated to a receiver by application of equation RF.3.

$$P_t = D_{xt} \times \sigma_t \quad \text{(RF.3)}$$

**Table A-3: Free Space Reflected Signal Variables**

| Symbol | Source | Description |
|---|---|---|
| $D_{xt}$ | Equation RF.2c | The power density at the target (t) of the signal that originated from the transmitter (x). |
| $P_t$ | Computed | The power created by the reflection of the incoming signal off of the target. |
| $\sigma_t$ | radar_signature of the target platform | The radar cross section of the target. |

### A.3.3 Reception of a free space signal

RF.4a is used for direct, one-way (communications, passive RF and jamming). RF.4b is used two-way (Radar, SAR).

$$P_r = D_{xr} \times \frac{\lambda^2}{4\pi} \times \frac{G_r}{L_r} \times F_{BW} \times F_{POL} \qquad \text{One-way, Transmitter-to-receiver} \qquad \text{(RF.4a)}$$

$$P_r = D_{tr} \times \frac{\lambda^2}{4\pi} \times \frac{G_r}{L_r} \times F_{40} \qquad \text{Two-way, Target-to-receiver} \qquad \text{(RF.4b)}$$

**Table A-4: Free Space Received Power Variables**

| Symbol | Source | Description |
|---|---|---|
| $F_{BW}$ | See section 2.5 | The fraction of the received signal that is admitted, accounting for possible mismatches in the frequency/bandwidth of the transmitted and the frequency/bandwidth of the receiver.<br><br>Note: this is not incorporated for radar interactions because it is assumed that the transmitter and receiver are matched. |
| $F_{POL}$ | transmitter polarization<br>receiver polarization<br>polarization_effects<br>antenna_pattern | The fraction of the received signal that is admitted, accounting for possible mismatches in the polarization of the transmitter and the receiver.<br><br>Note: This is not incorporated for radar interactions because it is assumed that the transmitter and receiver are matched. |
| $F_{40}$ | transmitter propagation_model | The pattern propagation factor. This accounts for the constructive/destructive interference between the direct and indirect signal paths.<br><br>Note: This is currently only implemented for radar interactions. |
| $D_{xr}$ | Equation RF.2b | The power density at the receiver of the signal that originated from the transmitter. |
| $D_{tr}$ | Equation RF.2d | The power density at the receiver of the signal that was reflected from the target. |
| $\lambda$ | transmitter wavelength | The wavelength of the transmitted signal. |
| $G_r$ | receiver antenna_pattern | The gain of the receiving antenna in the direction of the target object (receiver or platform). This includes any effects of electronic beam steering (Equation RF.6). |
| $L_r$ | receiver internal_loss | The internal losses in the receiver between the output of the antenna and the receiver. |
| $P_r$ | Computed | The received power. |

### A.3.4 Bandwidth ratio

The factor $F_{BW}$ is used to account for the fact that the frequency spectrum of the transmitter may not match the tuning band of the receiver. It is the fraction of the transmitter spectrum that is within the tuning band of the receiver.

$$F_{tl} = F_t - \frac{1}{2}B_t \quad \text{Lower frequency of transmitted spectrum}$$
$$F_{tu} = F_t + \frac{1}{2}B_t \quad \text{Upperfrequency of transmitted spectrum}$$
$$F_{rl} = F_r - \frac{1}{2}B_r \quad \text{Lower tuning frequency of the receiver}$$
$$F_{ru} = F_r + \frac{1}{2}B_r \quad \text{Upper tuning frequency of the receiver}$$

**Table A-5: Bandwidth Ratio Variables**

| Symbol | Source | Description |
|---|---|---|
| $B_r$ | receiver bandwidth | The bandwidth of the receiver. |
| $B_t$ | transmitter bandwidth | The bandwidth of the transmitter. |
| $F_r$ | receiver frequency | The center frequency of the range of frequencies the receiver can receive. |
| $F_t$ | transmitter frequency | The center frequency of the transmitter frequency spectrum. |

The resulting value of $F_{BW}$ depends on the relationship of the upper and lower frequencies of the transmitter and receiver.

$$F_{BW} = 0 \qquad \text{if } F_{xu} \le F_{rl}$$
$$F_{BW} = 0 \qquad \text{if } F_{xl} \ge F_{ru}$$
$$F_{BW} = \min\left(\frac{\min(F_{xu}, F_{ru}) - \max(F_{xl}, F_{rl})}{F_{xu} - F_{xl}}, 1.0\right) \qquad \text{(RF.5)}$$

### A.3.5 Receiver noise power

The following definitions apply to the computation of receiver noise power:

**Table A-6: Receiver Power Variables**

| Symbol | Source | Description |
|---|---|---|
| k | Internal constant | Boltzmann's constant (1.3806505E-23 J/deg-K). |
| B | receiver bandwidth -- or -- transmitter pulse_width | The bandwidth of the receiver. If the bandwidth was not specified AND if the transmitter is pulsed, the bandwidth will be computed as (1 / pulse_width) (i.e.: A matched filter will be assumed). |
| N | Computed | The noise power. |

| | | |
|---|---|---|
| NF | receiver noise_figure | The receiver noise figure (default 1.0). |
| $T_0$ | Internal constant | Standard temperature (290 deg-K). |
| $T_s$ | Computed | The systemnoise temperature. |

The noise power will be computed using the following process. The value from the first step whose conditions for use are satisfied will be used:

1. If noise_power was specified, used the defined value.

2. If the bandwidth cannot be determined, use the value of -160 dBW.

3. If ***noise_figure*** was specified and both ***antenna_ohmic_loss*** and ***receive_line_loss*** were omitted, compute the noise power as:

$$N = k \times T_0 \times B \times NF \quad \text{(RF.6a)}$$

4. Compute the noise power using the algorithm defined in "Radar Range Performance", Lamont V. Blake, 1986, Artech House, Inc., Chapter 4.

Noise temperature due to the antenna ($T_{ant}$ = sky temperature due to the antenna pointing angle):

$$T_a = T_0 + (0.876 \times T_{ant} - 254.0) / antenna\_ohmic\_loss \quad \text{(RF.6b)}$$

Noise temperature contribution due to receive line loss:

$$T_l = T_0 \times (receive\_line\_loss - 1.0) \quad \text{(RF.6c)}$$

Noise temperature contribution due to the receiver:

$$T_r = T_0 \times (noise\_figure - 1.0) \quad \text{(RF.6d)}$$

Total system temperature:

$$T_s = T_a + T_l + (receive\_line\_loss \times T_r) \quad \text{(RF.6e)}$$

Noise power:

$$N = k \times T_s \times B \quad \text{(RF.6f)}$$

### A.3.6 Antenna Gain Patterns

Each transmitter and receiver has associated with it an antenna gain pattern. Antenna patterns are created using the global ***antenna_pattern*** command. An antenna pattern is attached to a transmitter or receiver by using the ***antenna pattern*** command inside the ***transmitter*** and ***receiver***

block. If an antenna pattern has not been selected for a transmitter or receiver, a uniform gain of 1.0 will be assumed.

The gain pattern is a function of azimuth and elevation with respect to the pattern origin (typically the bore sight or pointing angle). For a given interaction, the azimuth and elevation of the point of interest with respect to the pattern origin is computed.

Antenna gain patterns may be represented in several ways:

- A rectangular table which provides the gain as a function of azimuth and elevation.
- A table.
- A uniform (constant) pattern.
- A circular sin(x)/x pattern.
- A rectangular sin(x)/x pattern.
- A cosecant pattern.
- A GENAP pattern (GENAP is a subset of the functionality provided by generalized antenna pattern routine found in the government TRAMS model).

Collections of tables may be used to form a composite pattern of polarization and frequency.

The gain of an electronically steered beam can be optionally modified to include the effects of pointing the beam at an angle off the normal of the array. This capability is enabled by using the ***electronic_beam_steering*** command in the ***transmitter*** or ***receiver***. The following equation is used:

$$G = G_0 \times \cos^n(\theta) \quad \text{RF.7}$$

**Table A-7: Antenna Gain Variables**

| Symbol | Source | Description |
|---|---|---|
| $G_0$ | antenna_pattern | The unmodified gain of the antenna when looking at the point of interest. |
| G | Computed | The gain, modified to include the effects of electronic beam steering. |
| θ | Computed | The angle between the normal to the antenna face and the vector to the point of interest. |
| N | electronic_beam_ steering_loss_ exponent | An optional exponent to reflect the amount of degradation of the gain as the beam is moved away from the normal of the antenna face. |

### A.3.7 Atmospheric Attenuation

Computation of atmospheric attenuation is enabled by the presence of the ***atmospheric_attenuation*** command inside the ***transmitter*** block.

There are currently two models available. These were extracted from Air Force models and **are currently only applicable to ground-based systems** (the tables assume the emitter is on the ground).

- An atmospheric absorption model written by L.V. Blake, Naval Research Laboratory. This is based on a family of 42 attenuation curves for frequencies between 100 MHz and 10 GHz and elevation angles between 0 and 10 degrees. The curves are flat beyond 300 nautical miles. These tables were published in 'Radar Systems Analysis', Section 15.1, David K. Barton, Artech Publishing.
- A collection of pre-computed tables that are valid for frequencies in the range 100 MHz to 18 GHz and 27 GHz to 40 GHz. Frequencies less than 100 MHz will assume 100 MHz. Frequencies between 18 GHz and 27 GHz and above 40 GHz will use a very computationally-intensive method to determine the attenuation and should be avoided.

There is another model in development based on the International Telecommunications Union (ITU) Recommendation ITU-R P.676. That implementation will work for air and surface platforms and support a wider range of frequencies.

### A.3.8 Propagation Algorithms

Computation of propagation effects (other than atmospheric attenuation) is enabled by the presence of the ***propagation_model*** command inside the ***transmitter*** block.

This currently supports one model:

- ***fast_multipath*** **-** An implementation of the method defined in 'Radar Range Performance Analysis', Lamont V. Blake, 1986, Artech House, Inc. It computes the effects of constructive or destructive interference due to the specular reflection of the signal off of a round, rough Earth. Two factors can be supplied to define the properties of the surface at the reflection point.
- ***ground_wave_propagation*** **-** A C++ port of the GRWAVE FORTRAN code available from the International Telecommunication Union, Radiocommunication Sector, Study Group 3 (Radiowave propagation).
- ***alarm*** **-** The Spherical Earth Knife Diffraction (SEKE) propagation code, ported to C++ from the Advanced Low-Altitude Radar Model (ALARM).

### A.3.9 Clutter Algorithms

The use of clutter is enabled by the presence of the ***clutter_model*** command in the ***receiver*** block. The options for clutter in baseline AFSIM are the following:

- ***surface_clutter*** **-** Model that utilizes radar characteristics, depression angle and backscatter coefficient lookup as functions of land form and land cover. Terrain is not utilized.
- ***alarm*** **-** Like ***surface_clutter,*** but taking terrain into account, this model is ported to C++ from the Advanced Low-Altitude Radar Model (ALARM).
- ***surface_clutter_table*** **-** represents clutter in a table that contains data as a function of target altitude and target range. Additionally, if the table is site-specific it will also contain data as a function of target bearing. The table may be provided directly, or it can be generated from sensor plot, using one of the other models. In the latter case, use of the table provides a speed optimization over using the other models directly.

Both the ***surface_clutter*** and ***alarm*** models are low angle clutter models, in that the clutter signal values they compute are only valid for small (positive and negative) depression angles. Because of this restriction they are typically only utilized for ground-sited radars. Both models draw from the same set of clutter backscatter coefficients, including the full set of data from the MIT/Lincoln Labs reference, "Low Angle Radar Clutter," by J. Barrie Billingsley.

## A.4 Radar Sensor (WSF_RADAR_SENSOR)

The AFSIM radar model effectively computes the power of a single pulse (or a continuous waveform) and then computes the effect of integrating multiple pulses.

### A.4.1 Calculation of Received Power

Applying equations RF.1 through RF.4, the following is used to calculate the received power from a single pulse (or a continuous waveform). Note that this does not including jamming. That is handled in a separate step.

$$
\begin{aligned}
P_r = \quad & D_{tr} \times \frac{\lambda^2}{4\pi} \times \frac{G_r}{L_r} \times F_{40} && \text{From RF.4b} \\
= \quad & P_t \times \frac{A_{tr}}{4\pi R_{tr}^2} \times \frac{\lambda^2}{4\pi} \times \frac{G_r}{L_r} \times F_{40} && \text{From RF.2d} \\
= \quad & D_{xt} \times \sigma_t \times \frac{A_{tr}}{4\pi R_{tr}^2} \times \frac{\lambda^2}{4\pi} \times \frac{G_r}{L_r} \times F_{40} && \text{From RF.3b} \\
= \quad & P_x \times \frac{A_{xt}}{4\pi R_{xt}^2} \times \sigma_t \times \frac{A_{tr}}{4\pi R_{tr}^2} \times \frac{\lambda^2}{4\pi} \times \frac{G_r}{L_r} \times F_{40} && \text{From RF.2c} \\
= \quad & P_{peak} \times DC \times \frac{G_x}{L_x} \times \frac{A_{xt}}{4\pi R_{xt}^2} \times \sigma_t \times \frac{A_{tr}}{4\pi R_{tr}^2} \times \frac{\lambda^2}{4\pi} \times \frac{G_r}{L_r} \times F_{40} && \text{From RF.1} \quad \text{(Radar.1)}
\end{aligned}
$$

### A.4.2 Signal Processing and Detection

The processed signal is computed as:

$$S = P_r \times PCR \times G_I \times AF \quad \text{(Radar.2)}$$

**Table A-8: Processed Power Variables**

| Symbol | Source | Description |
|---|---|---|
| AF | adjustment_factor | A general adjustment factor that can be used to account for other constant effects that are not provided by the model. |
| $G_i$ | integration_gain | The gain due to the integration of multiple pulses.<br>Note: This is computed internally if ***swerling_case*** is specified. |
| PCR | transmitter pulse_compression_ratio | The pulse compression ratio. |
| $P_r$ | Equation Radar.1 | The received power. |
| S | Computed | The processed power. |

The signal to noise is computed as:

$$SN = \frac{S}{N + C + J} \quad \text{(Radar.3)}$$

**Table A-9. Signal and Noise Variables**

| Symbol | Source | Description |
|---|---|---|
| C | Receiver clutter_model | The clutter power. |
| J | Equation Jam.1 | The incident jammer power. This is computed as the sum of the incident power on the radar receiver at the time of the detection interaction. |
| N | Equation RF.6 | The receiver noise power. |
| S | Equation Radar.2 | The processed power. |
| SN | Computed | The signal-to-noise (or interference) ratio. |

The detection of the target is determined by one of two mechanisms. A simplistic binary detector may be used by specifying a ***detection_threshold***. A successful detection is declared if the signal-to-noise exceeds this threshold.

A Marcum-Swerling detector may also be used, producing a probability of detection for a given signal-to-noise ratio. A successful detection is declared if the computed probability of detection exceeds the required probability of detection. This detector is selected by using the ***swerling_case, number_of_pulses_integrated, probability_of_false_alarm*** and ***detector_law*** commands.

## A.5 Passive RF Sensor (WSF_ESM_SENSOR)

Passive RF calculations (ESM, RWR) utilize the one-way equation.

The 'r' subscript values are for the passive RF receiver and the 'x' subscript values are for the sensor, jammer or communications transmitter. The expanded equation is as follows:

$$P_r = D_{xr} \times \frac{\lambda^2}{4\pi} \times \frac{G_r}{L_r} \times F_{BW} \times F_{POL} \quad \text{From RF.4a}$$

$$= P_x \times \frac{A_{xr}}{4\pi R_{xr}^2} \times \frac{\lambda^2}{4\pi} \times \frac{G_r}{L_r} \times F_{BW} \times F_{POL} \quad \text{From RF.2b}$$

$$= P_{peak} \times DC \times \frac{G_x}{L_x} \times \frac{A_{xr}}{4\pi R_{xr}^2} \times \frac{\lambda^2}{4\pi} \times \frac{G_r}{L_r} \times F_{BW} \times F_{POL} \quad \text{From RF.1} \quad \text{(ESM.1)}$$

The signal-to-noise is computed as:

$$SN = \frac{P_r}{N} \quad \text{(ESM.2)}$$

**Table A-10. Passive Receiver Noise Variables**

| Symbol | Source | Description |
|---|---|---|
| N | Equation RF.6 | The receiver noise power. |
| $P_r$ | Equation ESM.1 | The processed power. |
| SN | Computed | The signal-to-noise (or interference) ratio. |

A successful detection will be declared if *SN* exceeds the threshold defined by:

- The value of ***pulsed_detection_threshold*** if the transmitted signal is pulsed
- The value of ***continuous_detection_threshold*** if the transmitted signal was non-pulsed
- The value of ***detection_threshold*** if neither of the above thresholds was specified

## A.6 SAR Sensor (WSF_SAR_SENSOR)

SAR calculations are an extension of the radar calculations.

### A.6.1 Required Collection Time

The equation used to compute the time required to collect an image of the desired resolution is:

$$T_{OT} = \frac{KR_s\lambda}{2d_A V_G |\sin(\alpha)|\cos(\delta)} \qquad \text{(SAR.1)}$$

**Table A- 11. SAR Collection Time Variables**

| Symbol | Source | Description |
|---|---|---|
| $d_A$ | Computed | The desired azimuth resolution. |
| K | doppler_overcollect_ratio | The overcollect ratio (default 1.0). |
| $R_s$ | Computed | The slant range from the sensor to the image center. |
| $V_G$ | Computed | The ground speed of the sensing platform. |
| λ | transmitter frequency –or- wavelength | The frequency of the transmitted signal. |
| α | Computed | The azimuth angle between the ground track of the sensing platform and the vector to the image center. |
| δ | Computed | The azimuth angle between the ground. |

## A.7 RF Jammer (WSF_RF_JAMMER)

Jamming calculations utilize the one-way equation where the transmitter is the jammer and the receiver is a radar or communications receiver. Jamming calculations take place at the time a radar detection or communication attempt occurs. AFSIM will sum the power for every possible jammer than can affect the output (i.e. if there is in-band power that would affect the receiver).

The 'r' subscript values are for the sensor or communications receiver and the 'x' values are for the jamming transmitter. The expanded equation is a follows:

$$P_r = D_{xr} \times \frac{\lambda^2}{4\pi} \times \frac{G_r}{L_r} \times F_{BW} \times F_{POL} \qquad \text{From RF.4a}$$

$$= P_x \times \frac{A_{xr}}{4\pi R_{xr}^2} \times \frac{\lambda^2}{4\pi} \times \frac{G_r}{L_r} \times F_{BW} \times F_{POL} \qquad \text{From RF.2b}$$

$$= P_{peak} \times DC \times \frac{G_x}{L_x} \times \frac{A_{xr}}{4\pi R_{xr}^2} \times \frac{\lambda^2}{4\pi} \times \frac{G_r}{L_r} \times F_{BW} \times F_{POL} \qquad \text{From RF.1} \qquad \text{(Jam.1)}$$

## A.8 Communications (WSF_RADIO_TRANSCEIVER)

Communications calculations use the one-way equation.

The 'r' subscript values are for the communications receiver and the 'x' values are for the communications transmitter. The expanded equation is a follows:

$$P_r = D_{xr} \times \frac{\lambda^2}{4\pi} \times \frac{G_r}{L_r} \times F_{BW} \times F_{POL} \qquad \text{From RF.4a}$$

$$= P_x \times \frac{A_{xr}}{4\pi R_{xr}^2} \times \frac{\lambda^2}{4\pi} \times \frac{G_r}{L_r} \times F_{BW} \times F_{POL} \qquad \text{From RF.2b}$$

$$= P_{peak} \times DC \times \frac{G_x}{L_x} \times \frac{A_{xr}}{4\pi R_{xr}^2} \times \frac{\lambda^2}{4\pi} \times \frac{G_r}{L_r} \times F_{BW} \times F_{POL} \qquad \text{From RF.1} \quad \text{(Comm.1)}$$

The signal-to-noise ratio is computed as:

$$SN = \frac{P_r}{N + J} \qquad \text{(Comm.2)}$$

**Table A-12: Communications Signal and Noise Variables**

| Symbol | Source | Description |
|---|---|---|
| J | Equation Jam.1 | The incident jammer power. This is computed as the sum of the incident power on the receiver at the time of the interaction. |
| N | Equation RF.6 | The receiver noise power. |
| $P_r$ | Equation Comm.1 | The processed power. |
| SN | Computed | The signal-to-noise (or interference) ratio. |

The communications attempt will be declared successful if *SN* exceeds the ***detection_threshold*** for the ***receiver***.

## A.9 IRST Sensor (WSF_IRST_SENSOR)

### A.9.1 Computing the target irradiance

Determine the background radiance. This includes a relatively simply capability to include the effects of looking up against the sky or down at the ground.

Compute the contrast radiant intensity;

$$I_c = I_s - L_{bkg} \times A_{proj}$$

**Table A-13: IRST Contrast Radiant Intensity Variables**

| Symbol | Source | Description |
|---|---|---|
| $I_s$ | platform infrared_signature | The source radiant intensity (infrared radiant intensity) of the target. |
| $L_{bkg}$ | | The background radiant intensity. |
| $A_{proj}$ | platform optical_signature | The projected area of the target as seen by the sensor. |
| $I_c$ | Computed | The contrast radiant intensity of the target. |

Compute the atmospheric transmittance, $t$ (the fraction of the signal that remains after propagation along the path):

Compute the effective target irradiance, $E_{eff}$ (sometimes known as CEI)

$$E_{eff} = \frac{t \times I_c}{R^2}$$

### A.9.2 Adjusting for installation effects

Sensors are often mounted behind a window, which will mask regions from the field of view, or otherwise reduce the signal. This masking or signal reduction is collectively called 'installation effects', and is accounted for by the use of an ***antenna_pattern*** command in the ***receiver*** block (while there is no 'antenna' in an infrared sensor, it is being treated as such for convenience). The command should refer to an antenna gain pattern where the gain (or more possibly, the loss) represents a factor, by which the effective target irradiance should be modified to account for installation effects, i.e.:

$$E'_{eff} = E_{eff} \times G \qquad \text{(IRST.1)}$$

where $G$ is the 'antenna gain' in the direction of interest. Setting the gain to a very small value in the regions that are outside the window effectively makes targets in that region undetectable.

### A.9.3 Computing the probability of detection

The probability of detection is computed using the following equation:

$$SN = \frac{E'_{eff}}{NEI}$$
$$\beta = SN - S_{thresh}$$
$$P_d = 1 - Q(\beta)$$

**Table A-14: IRST Probability of Detection Variables**

| Symbol | Source | Description |
|---|---|---|
| $E'_{eff}$ | Equation IRST.1 | The effective target irradiance. |
| NEI | noise_equivalent_ irradiance | The equivalent irradiance of the noise of the sensor. |
| $P_d$ | Computed | The probability of detection. |
| Q(β) | | The Gaussian probability function (see the 'Handbook of Mathematic Functions', Abramowitz and Stegun, equation 26.2.5). |
| SN | Computed | Signal-to-noise ratio. |
| $S_{thresh}$ | detection_threshold | The detection threshold. |

# Appendix B- AFSIM Synthetic Aperture Radar Equations

## B.1 The Radar Range Equation

The following equation is the standard equation used by AFSIM to compute the received power from a single pulse of a radio frequency signal that is transmitted, reflects off an object and is then received. This does not assume the transmitter and receiver are co-located, and does not account for any additional signal processing techniques such as pulse compression or integration of multiple pulses:

$$P_r = P_{peak}(DC)\left(\frac{G_x}{L_x}\right)\left(\frac{A_{xt}}{4\pi R_{xt}^2}\right)\sigma\left(\frac{A_{tr}}{4\pi R_{tr}^2}\right)\left(\frac{\lambda^2}{4\pi}\right)\left(\frac{G_r}{L_r}\right)F_{40}\,F_{BW}\,F_{POL}$$

where:

**Table B-1: Radar Range Equation Variables**

| Variable | AFSIM source | Description |
|---|---|---|
| $P_{peak}$ | transmitter power | The peak transmitted power. |
| DC | transmitter duty_cycle | The user defined duty-cycle of the transmitter (default: 1.0, if not defined). |
| $\lambda$ | transmitter frequency or wavelength | The wavelength of the radiated signal. |
| $G_x, G_r$ | transmitter / receiver antenna_pattern | The gain of the transmitter and receiver antennas. |
| $L_x, L_r$ | transmitter / receiver internal_loss | The internal losses within the transmitter and receiver. |
| $A_{xt}, A_{tr}$ | transmitter attenuation_model | The one-way atmospheric attenuation factor (0..1) computed by the attenuation mode. |
| $R_{xt}, R_{tr}$ | Computed | The range from the transmitter-to-target and target-to-receiver. |
| $\sigma$ | | The radar cross section of the target (which could be the target or a 'resolution cell'). |
| $F_{40}$ | transmitter propagation_model | The pattern propagation factor that accounts for the constructive and destructive interference between the direct and indirect reflections. |
| $F_{BW}$ | | The factor that accounts for bandwidth mismatches between the bandwidth of the transmitted signal and the bandwidth of the receiver. This is primarily for interactions between radar transmitters and passive RF receivers, or jammers and radar receivers. It is not intended to capture the effects of non-ideal matched filters in a system. |
| $F_{pol}$ | transmitter / receiver polarization | The factor that accounts for the polarization mismatch of the transmitted signal and the receive antenna. |

For a SAR, the transmitter and receiver are co-located, so $R = R_{xt} = R_{tr}$ and A = $A_{xt} = A_{tr}$. Also, the gain of the transmit and receive antennas will be assumed identical, so $G = G_t = G_r$. Furthermore, the following will be assumed:

- There are no indirect signals to interfere with the main signal. Therefore $F_{40} = 1$.

- The bandwidth of the receiver has been established to capture the full bandwidth of the transmitted signal. Therefore, $F_{BW} = 1$.
- The polarization of the received signal is the same as the polarization of the transmitted signal. Therefore, $F_{POL} = 1$.
- If we also define the total atmospheric attenuation loss as:

$$L_{atm} = \frac{1}{A_{xt} A_{tr}} = \frac{1}{A^2}$$

With the above assumptions, we get the familiar equation for power received from a single pulse, where the transmitter and receiver are co-located:

$$P_r = P_{peak} \frac{G^2 \lambda^2 \sigma}{(4\pi)^3 R^4 L_x L_r L_{atm}}$$

This is the same as equation (1) in reference 1, noting that:

$$L_{radar} = L_x L_r$$

and that equation (8) has been used to replace $A_e$:

$$A_e = \frac{G_A \lambda^2}{4\pi}$$

Further note that reference 1 goes on to represent the effective aperture as a product of the actual aperture area multiplied by an 'aperture efficiency'. We will not perform that step here and assume that any aperture efficiency has been represented in the AFSIM antenna_pattern.

The received signal must compete with the noise present within the system. AFSIM computes noise power using the following:

**Table B-2: Receiver Noise Variables**

| Symbol | AFSIM source | Description |
|---|---|---|
| $k$ | Internal constant | Boltzmann's constant (1.3806505E-23 J/deg-K). |
| $B_N$ | receiver bandwidth | The bandwidth of the receiver. If the bandwidth was not specified AND if the transmitter pulse_width is specified, the bandwidth will be computed as (1 / pulse_width) (i.e.: a matched filter will be assumed. |
| $F_N$ | receiver noise_figure | The receiver noise figure (default 1.0). |
| $N$ | Computed | The noise power. |
| $T_0$ | Internal constant | The standard temperature (290 deg-K). |

The receiver noise at the antenna port is computed as:

$$N = kT_0 B_N F_N$$

(Note: Section 2.6 in reference 2 describes other forms of computing the noise power, but these are primarily for surface-based systems.)

The signal-to-noise ratio for a single pulse at the antenna port is then:

$$SNR_{ant} = \frac{P_r}{N} = P_{peak} \frac{G^2 \lambda^2 \sigma}{(4\pi)^3 R^4 L_x L_r L_{atm}} \frac{1}{kT_0 B_N F_N}$$

(This is the same as equation (5) in reference 1, with the substitutions noted above.)
A SAR utilizes two signal processing techniques to increase the effective SNR in the image.

- $G_a$ = SNR gain due to azimuth processing (coherent pulse integration)
- $G_r$ = SNR gain due to range processing (pulse compression)

This will result in the signal-to-noise ratio a target within the image to be:

$$SNR_{image} = SNR_{ant} G_a G_r = \frac{P_r}{N} = P_{peak} \frac{G^2 \lambda^2 \sigma}{(4\pi)^3 R^4 L_x L_r L_{atm}} \frac{1}{kT_0 B_N F_N} G_a G_r$$

(This is the same as equation (11) in reference 1, with the substitutions noted above.)

## B.2 Azimuth Processing Gain (Coherent Pulse Integration)

The creation of a SAR image involves the collection of a large number of pulses coherently over some duration of time that is suitable for producing an image of the desired quality.

**Table B-3: SAR Dwell Time Variables**

| Variable | AFSIM source | Description |
|---|---|---|
| $f_p$ | transmitter pulse_repetition_frequency | The pulse repetition frequency. |
| $K_a$ | doppler_filter_broadening_factor | |
| $K_d$ | doppler_foldover_margin_factor | |
| $t_D$ | dwell_time or computed | The dwell time, or image collection time. |
| $\delta_a$ | resolution or computed | The desired azimuth resolution. |
| $V$ | | The vehicle velocity vector. |
| $\theta_{sq}$ | computed | The 'squint angle', defined as the angle between the velocity vector and the line-of-sight vector to the center of the image area.<br><br>**Note**: In some documents this will be measured as the angle-of-broadside, causing the use sin() and cos() to be reversed. |
| $n_{image}$ | Computed | The total number of pulses collected in forming the image. |

Equation (5) from Reference 3 is used to compute the dwell time from the desired cross range/azimuth resolution:

$$t_D = \frac{\lambda K_a R}{2V \delta_a \sin \theta_{sq}}$$

Note that AFSIM lets the user specify either the desired resolution or dwell time. In the latter case, AFSIM will use the above equation to solve for the achievable resolution given the dwell time.

The azimuth gain is the total number of pulses collected, which is just the collection time times the pulse repetition frequency:

$$G_a = n_{image} = t_D f_p = \frac{\lambda K_a R f_p}{2V\delta_a \sin\theta_{sq}}$$

## B.3 Range Processing Gain (Pulse Compression)

**Table B-4: Range Processing Gain Variables**

| Variable | AFSIM source | Description |
|---|---|---|
| $\tau_u$ | transmitter pulse_width | The uncompressed pulse width. |
| $\frac{\tau_u}{\tau_c}$ | transmitter pulse_compression_ratio | The pulse compression ratio. |

The range processing gain due to pulse compression is:

$$G_r = \frac{\tau_u}{\tau_c}$$

## B.4 Various Forms Of The Signal-To-Noise Equation

Substituting the results for $G_a$ and $G_r$ into the equation for $SNR_{image}$:

$$SNR_{image} = P_{peak}\frac{G^2\lambda^2\sigma}{(4\pi)^3 R^4 L_x L_r L_{atm}}\frac{1}{kT_0 B_N F_N} G_a G_r$$

$$SNR_{image} = P_{peak}\frac{G^2\lambda^2\sigma}{(4\pi)^3 R^4 L_x L_r L_{atm}}\frac{1}{kT_0 B_N F_N}\frac{\lambda K_a R f_p}{2V\delta_a \sin\theta_{sq}}\frac{\tau_u}{\tau_c}$$

This is the form used by AFSIM to compute the return from an object with a radar cross section of $\sigma$. This could be a target or a resolution cell.

Additional forms of the equation are often seen in the literature. The remainder of this section will show how the above equation is equivalent.

In the case of a matched filter:

$$B_N = \frac{1}{\tau_c}$$

Substituting:

$$SNR_{image} = P_{peak}\frac{G^2\lambda^3\sigma}{(4\pi)^3 R^3 L_x L_r L_{atm}}\frac{\tau_c}{kT_0 B_N F_N}\frac{K_a f_p}{2V\delta_a \sin\theta_{sq}}\frac{\tau_u}{\tau_c}$$

$$SNR_{image} = P_{peak}\tau_u f_p \frac{G^2\lambda^3\sigma}{(4\pi)^3 R^3 L_x L_r L_{atm}}\frac{1}{kT_0 F_N}\frac{K_a}{2V\delta_a \sin\theta_{sq}}$$

$$SNR_{image} = P_{avg}\frac{G^2\lambda^3\sigma}{(4\pi)^3 R^3 L_x L_r L_{atm}}\frac{1}{kT_0 F_N}\frac{K_a}{2V\delta_a \sin\theta_{sq}}$$

Where average power is defined to be:

$$P_{avg} = P_{peak}\tau_u f_p$$

One form of interest is when the target is a bare resolution cell (i.e.: the ground). This is sometimes called the 'clutter-to-noise ratio', or CNR. Equation (23) of reference 1 defines the area of the resolution cell as:

$$\sigma = \sigma^0 \delta_a \delta_{rg} = \sigma^0 \delta_a \frac{\delta_r}{\cos \psi_g}$$

where:

**Table B-5: Resolution Cell Variables**

| Variable | AFSIM source | Description |
|---|---|---|
| $\sigma^0$ | backscatter_coefficient | The backscatter coefficient. |
| $\delta_r$ | | Range resolution (as computed from the effective pulse width). |
| $\delta_{rg}$ | | Range resolution in the ground plane. |
| $\psi_g$ | | Grazing angle. The angle between the line-of-sight vector and the plane tangent to the surface at the point being viewed. |

Substituting:

$$SNR_{image} = P_{avg} \frac{G^2 \lambda^3}{(4\pi)^3 R^3 L_x L_r L_{atm}} \sigma^0 \delta_a \frac{\delta_r}{\cos \psi_g} \frac{1}{kT_0 F_N} \frac{K_a}{2V \delta_a \sin \theta_{sq}}$$

$$SNR_{image} = P_{avg} \frac{G^2 \lambda^3}{(4\pi)^3 R^3 L_x L_r L_{atm}} \frac{\sigma^0 \delta_r}{\cos \psi_g} \frac{1}{kT_0 F_N} \frac{K_a}{2V \sin \theta_{sq}}$$

which is basically equivalent to the myriad forms presented in Appendix B of reference 1 (however they always assumed broadside collection, so $\sin \theta_{sq}$ was always 1).

## B.5 Creation of AFSIM Pseudo-Images

AFSIM does not produce true images, but rather produces pseudo-images that indicate the objects that are in the image, the number of resolution cells (pixels) occupied by the object, and the intensity of the object.

- The user cues the system to the desired location and turns the system on. The model constructs a list of the targets that could potentially be in the image. The target list will encompass targets that are slightly outside the image region in order to account for the fact that a target may move into the image.
- At periodic intervals (defined by 'frame_time', default of 1 second), the model computes and accumulates data for each of the targets from step 1. The results of these detection results will be accumulated, much as a SAR accumulates pulses. If the target is obscured by terrain during a given sample, it will not have any contributing pulses defined during that interval.
- At some point, the SAR will be turned off. At that point the model will take the accumulated results and produce the pseudo-image (WsfImage) and send a message containing the image (WsfImageMessage) to those who have subscribed.

The following variables will be used in the following section:

**Table B-6: Pseudo-Image Generation Variables**

| Variable | AFSIM source | Description |
|---|---|---|
| $t_F$ | frame_time | The update interval between samples when forming the image. |
| $n_{actual}$ | | The actual number of pulses integrated. This may be different from $n_{image}$ if the sensor was turned off before or after the required time. |
| $t_{sample}$ | | The length of the sample. This will be $t_F$ for all but the final sample. |
| $n_{sample}$ | | The number of pulses received during a sample. |
| $P_{sample}$ | | The received signal from a specific target during a sample. |
| $NP_{sample}$ | | The number of resolution cells (pixels) covered by a specific target during a sample. |
| $\sigma_{opt}$ | optical_signature | The optical signature of the target. |
| $P_{sum}$ | | The sum of the sampled received signals for a specific target. |
| $NP_{sum}$ | | The sum of the sampled pixel counts for a specific target. |
| $N_{seen}$ | | The number of samples in which a specific target was visible (not obscured by the terrain). |
| $P_{ref}$ | | The reference signal that corresponds to a zero intensity in the output image. This would typically be minimum clutter-to-noise ratio. |
| $P_{range}$ | | A normalizing value used to scale the received signals into a range of [0..1]. |
| $CNR$ | | The expected signal-to-noise ratio of a return from single resolution cell. |
| $CNR_{min}$ | detection_threshold | The minimum acceptable value for CNR for an image to be declared. |

Step 1 computes the anticipated dwell time ($t_D$) and the number of pulses to be collected ($n_{image}$). In addition, it calculates the anticipated value of $CNR$.

For each sample in step 2, the number of pulses received during the sample interval is:

$$n_{sample} = f_p t_{sample}$$

The number of resolution cells (pixels) occupied by the target for a given sample is the projected area of the target (optical cross section) divided by the size of a resolution cell:

$$NP_{sample} = \frac{\sigma_{opt}}{\delta_a \delta_r}$$

The received power per resolution cell from the target during the sample interval is:

$$P_{sample} = \frac{SNR_{image}}{N} \frac{n_{sample}}{n_{image}} \frac{1}{NP_{sample}}$$

Note that the noise has been removed from the accumulation. This must be done to account for the possibility that in some samples the target may not be visible, or that the actual dwell time may be longer or shorter than what was initially computed. The other terms account for the fact

that the internal routine that calculates $SNR\_image$ computes the return for the entire target for the expected dwell time.

For each sample of a target in which the target is not obscured by terrain, the following is performed:

$$N_{seen} = N_{seen} + 1$$
$$P_{sum} = P_{sum} + P_{sample}$$
$$NP_{sum} = NP_{sum} + NP_{sample}$$

In step 3, the achieved clutter-to-noise ratio must be computed. It is done at this point because the actual number of pulses collected is now known (the user may choose to turn the system off before or after the time required).

$$CNR_{actual} = CNR\frac{n_{actual}}{n_{image}}$$

If $CNR_{actual}$ is greater than or equal to $CNR_{min}$, the image will be declared acceptable and will contain the targets as processed below. If image is declared unacceptable, the image will be produced with no targets.

The reference signal level will be defined to be:

$$P_{ref} = \frac{CNR_{actual}}{N}$$

If the image is declared acceptable, the following will be produced for each target:

The number of pixels (resolution cells) occupied by the target. This will just be the average of the pixel counts from each sample where there target was not obscured by terrain:

$$NP = \frac{NP_{sum}}{N_{seen}}$$

The intensity of the pixel is then computed. The integrated return from a resolution cell (aka, clutter cell) represents the 'zero' intensity, or the return that will return a pixel value of zero.

$$I = \frac{P_{sum} - P_{ref}}{P_{range}}$$

A value less than zero is clamped to zero, while values greater than one are clamped to one.

## B.6 References


1. Doerry, Armin W., "Performance Limits for Synthetic Aperture Radar - second edition", Sandia National Laboratory Report SAND2006-0821, 2006.
2. Johnson, Jeffery, "AFSIM Communications, Sensor and Jamming Equations", Boeing
3. Renaud, Matthew J., "Synthetic Aperture Radar Mode Constraints", Boeing, 2007
4. Skolnik, M., **Radar Handbook**, 2nd edition, New York: McGraw-Hill, 1990.

# Appendix C- AFSIM Electronic Warfare Overview

## C.1 Electronic Warfare Architecture

The Electronic Warfare (EW) capability in AFSIM provides for the effect(s) of an EW technique from existing data, lower-level models (engineering models, engagement models, etc.) to be modeled without having to capture a lot of high-fidelity information (i.e. pulse level modeling). This results in primarily data/table driven capabilities, and the use of equations where behavior is well defined, documented, and proven to follow specific guidelines. The benefits of this level of modeling provide improved runtime, and allows for less detail on the many interactions of different effects associated with techniques on different systems to be evaluated using single/few model definitions. The cons are a possible loss in fidelity and some edge cases that may not always be captured properly.

The general coding approach to implementing the EW techniques in AFSIM was to follow an Object-Oriented (OO) approach/framework allowing for easier addition/update of new EW effects, while adding some complexity on the positioning of data and methods within the architecture. The EW architecture was further divided into Electronic Attack (EA) and Electronic Protect (EP) classes, with ES being considered in another part of the AFSIM architecture. Each of the EA and EP classes has multiple techniques, each with their own predefined and user-defined characteristics known as effects that the technique supplies as shown in Figure C-1.

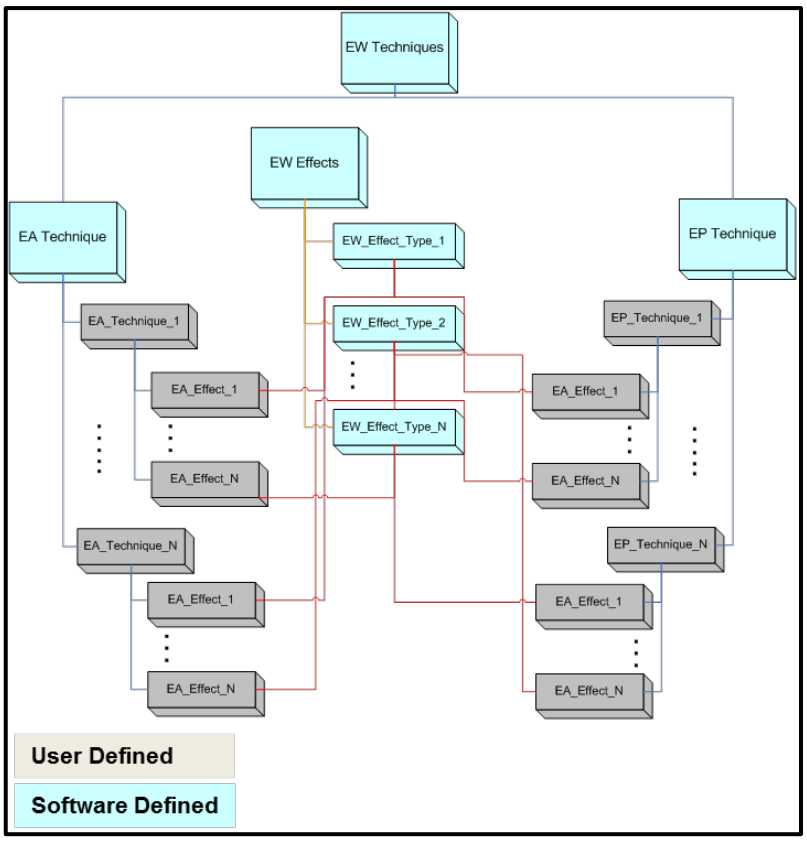


**Figure C-1. EW Techniques Architecture**

EA techniques are associated with transmitting systems within the AFSIM definition of a system as shown in Figure C-2. Likewise, the EP techniques are associated with receiving systems as shown in Figure C-3. The transmitter can then deliver the EA techniques to a receiver upon a successful interaction between the transmitter and receiver. The interaction between the EA & EP techniques and their associated effects are accomplished through user mappings of the EA & EP interactions and occur during a transmitter-receiver interaction as depicted in Figure C-3. Although Figure C-2 depicts a sensor, the interaction of EW techniques is also pertinent to communications systems in AFSIM, as they share the same base receiver functionality.

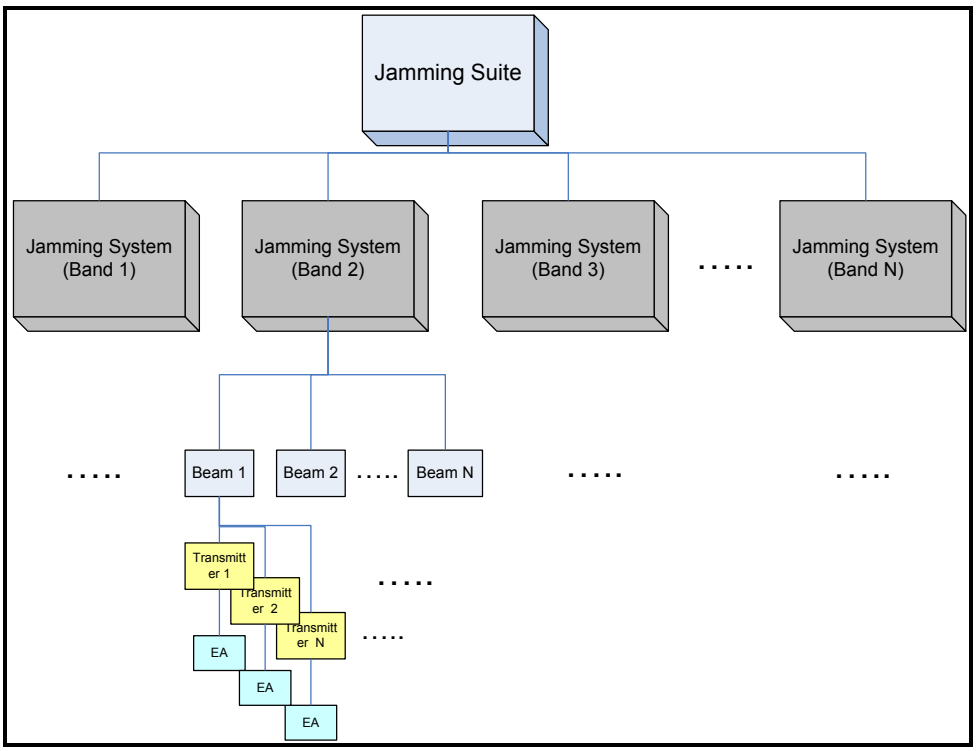


**Figure C-2. Jamming System Architecture in AFSIM Showing EA Techniques**

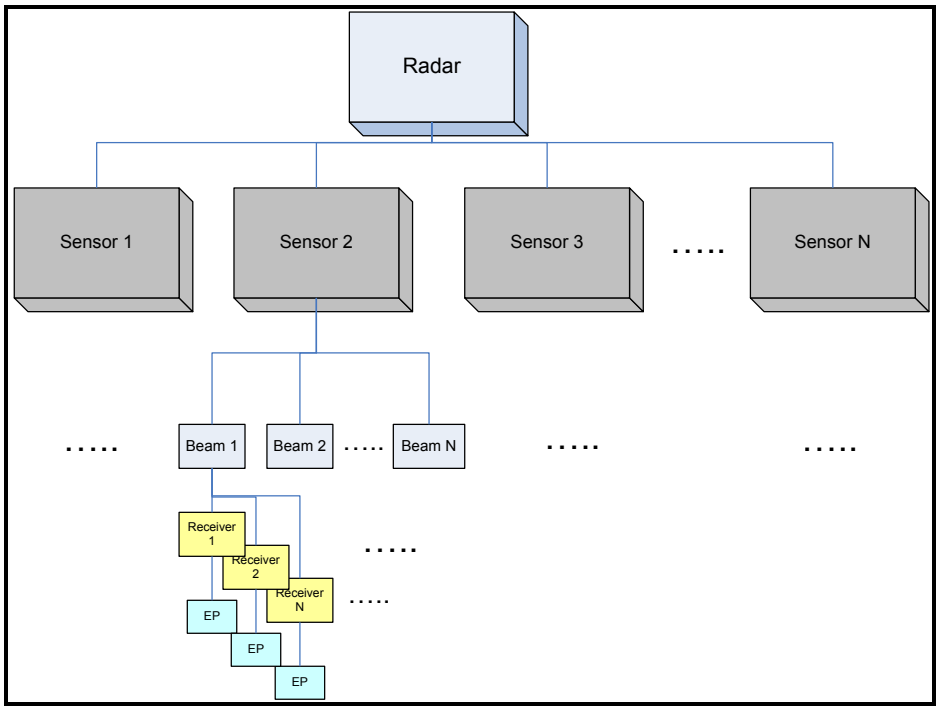


**Figure C-3. Radar System Architecture in AFSIM Showing EP techniques**

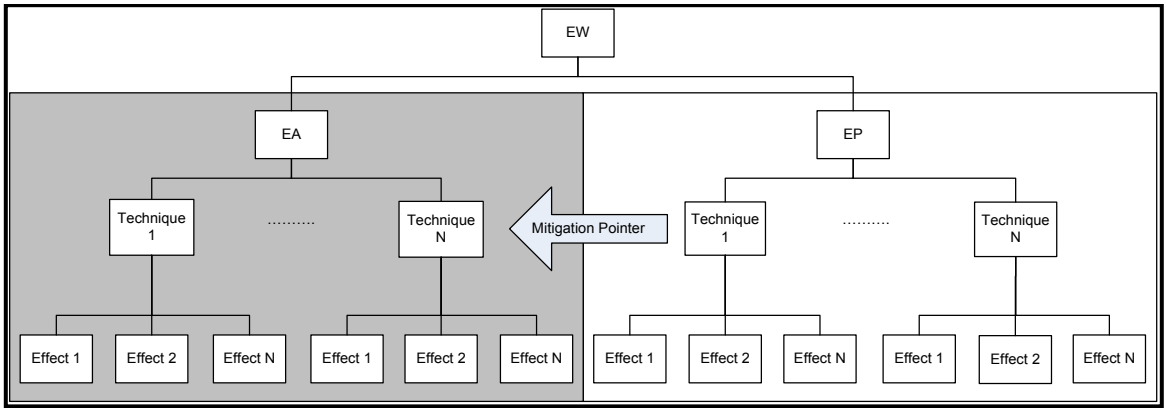


**Figure C-4. Mapping of EA and EP Interactions in the AFSIM EW Architecture**

## C.2 Electronic Attack Effects

EA specific effects in AFSIM that are defined are as follows with a short description:

- WSF_COMM_EFFECT
  - Induce communication effects
- WSF_COVER_PULSE_EFFECT
  - Induce cover pulse effects with probabilities
- WSF_FALSE_TARGET_EFFECT
  - False-Target effect with track creation
  - Pulse type effect
- WSF_POL_MOD_EFFECT
  - Derives from the WSF_SLC_DEGRADE_EFFECT with the addition of some data to model a polarization modulation technique
  - Induces a degradation on the EP SLC effect
- WSF_POWER_EFFECT
  - Jammer gain/degrade effect
  - Base type for most effects
- WSF_PULSE_EFFECT
  - Pulse level effects
  - Base type for other pulse level base types
- WSF_RADIUS_EFFECT
  - Target/Jammer/Radar trio radius effects
- WSF_REPEATER_EFFECT
  - Application of repeater jamming effects
- WSF_RPJ_EFFECT
  - Random Pulse Jamming/ Modulation (RPJ/RPM) effect
  - Pulse type effect
- WSF_SIMPLE_FT_EFFECT
  - Simple False-Target effect w/o track creation
  - Pulse type effect
- WSF_SLC_DEGRADE_EFFECT
  - Extra SLC degradation factors
- WSF_TRACK_EFFECT
  - Induce track error effect(s)

These EA effects also utilize inheritance from base classes within AFSIM, as depicted in Figure C-5, that are also able to have their input commands used within the inheriting class from a command input level.

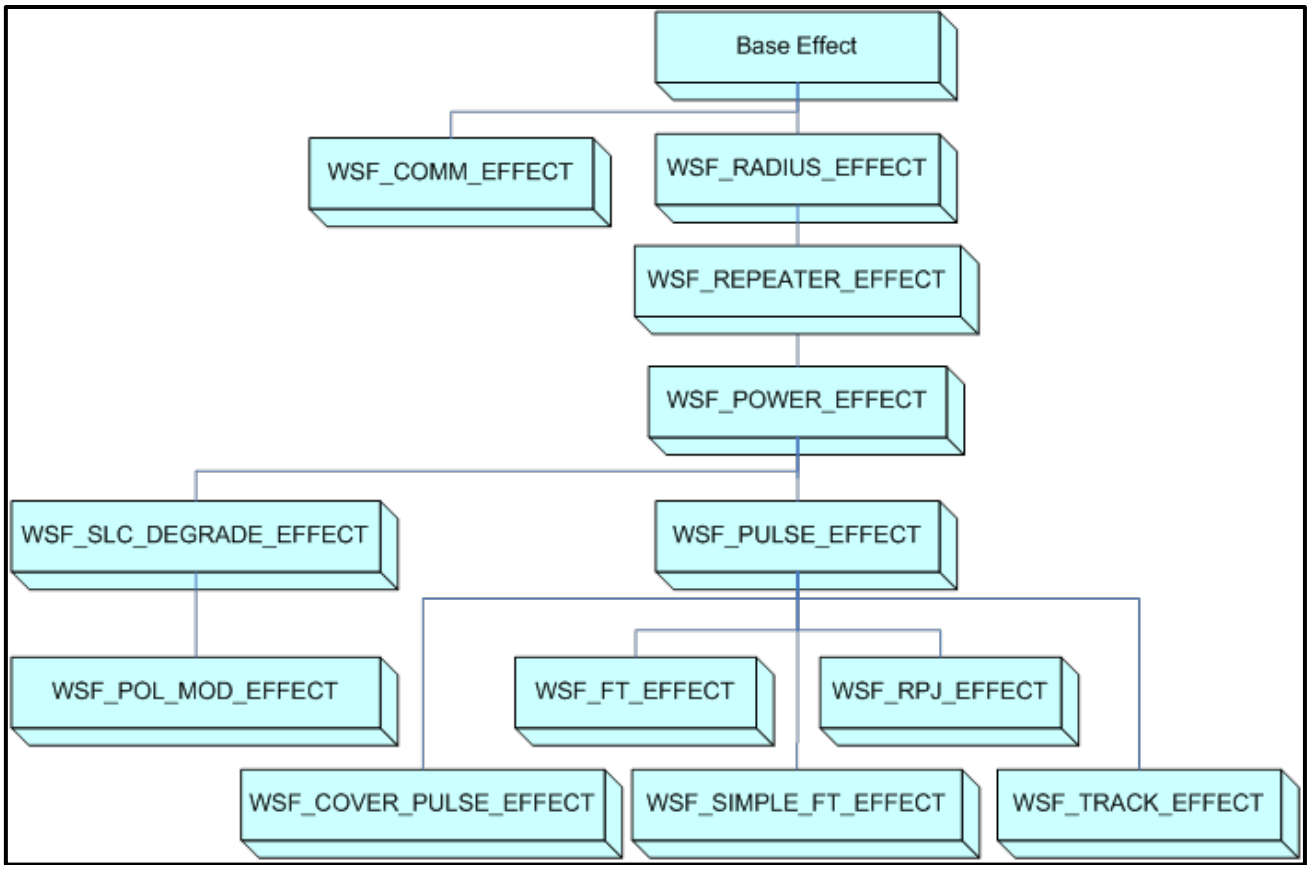


**Figure C-5. Hierarchy of Base Type Effects for EA**

## C.3 Electronic Protect Effects

EP specific effects in AFSIM that are defined are as follows with a short description:

- WSF_AGILITY_EFFECT
  - Frequency and/or mode agility/diversity effects
- WSF_COMM_EFFECT
  - Mitigate communication effects
- WSF_POWER_EFFECT
  - Jammer gain/degrade effect
  - Base type for most effects
- WSF_PULSE_EFFECT
  - Pulse level effects
  - Base type for other pulse type effects
- WSF_PULSE_SUPPRESS_EFFECT
  - Pulse Suppression effect for pulse type EA effects
  - Pulse type effect
- WSF_SLB_EFFECT
  - Sidelobe Blanker effects
- WSF_SLC_EFFECT
  - Sidelobe Canceller effects

- WSF_TRACK_EFFECT
    - Mitigate track error effects

These EP techniques also use inheritance from base classes within AFSIM, as depicted in Figure C-6, that are also able to have their input commands executed within the inheriting class from a command input level.

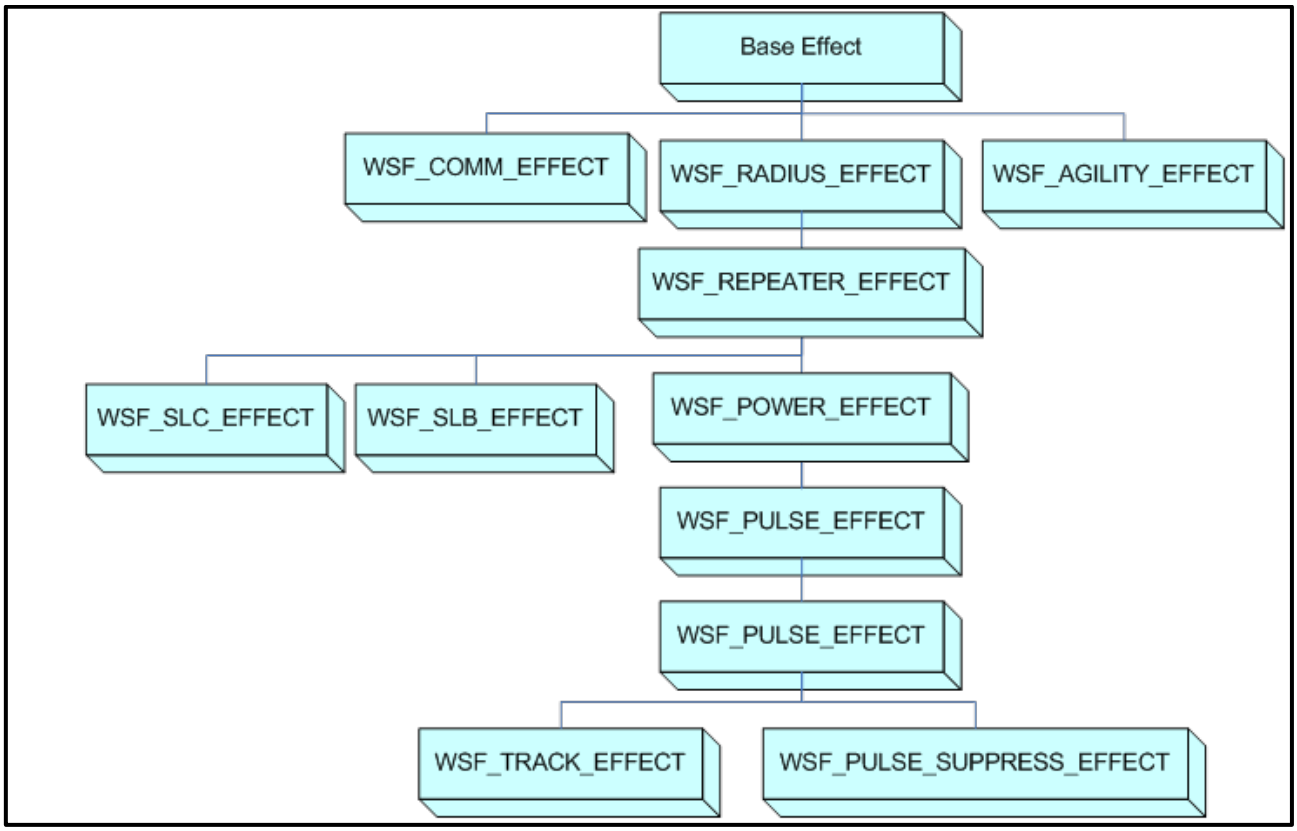


**Figure C-6. Hierarchy of Base Type Effects for EP**

## C.4 EW Interaction Overview

As previously described, the EW effects in AFSIM include EA effects and EP effects that are applied during a transmitter-receiver or transmitter-target-receiver interaction as defined by this effects behavior in software and user modifiable inputs. During this interaction, the interaction data for the EW are applied by using the unmitigated EA effect, then applying any mitigating EP effects to the EA effect. This process is then repeated until all EA effects on all techniques are applied along with mitigating EP effects, and aggregated into a single data set that is used to modify the interaction between the transmitter-receiver and transmitter-target-receiver. The aggregation of this EW effects is further defined below for more insight into how AFSIM is applying these effects and is graphically depicted in Figure C-7.

### C.4.1 Electronic Attack Effect Coherency

Jamming power within an interaction calculation is calculated using general AFSIM Electromagnetic calculations. Each EA effect possesses one or more of the coherency types on the

individual effect. As the effects are applied, the types of effect coherencies encountered are summed, and at the end of all the EW effects calculations the jamming power is divided into three types of jamming power (non-coherent, non-coherent-pulse, coherent) based on the EA effects coherencies encountered during the application of the effects. The coherency types available are defined in the table below along with the jamming power type it is summed into.

**Table C-1. Effect Coherency Types**

| Effect Coherency Types | Description | Mapped Jamming Power Type |
|---|---|---|
| None | Coherency not specified for the given effect. Assumes Non-Coherent. | Noise |
| Non-Coherent | Waveform is non-coherent with the transmit and/or expected receive waveform. Assumes continuous noise type waveform in most basic sense. | Noise |
| Non-Coherent Pulse | Waveform is pulsed and is non-coherent with the transmit and/or expected receive waveform. Assumes pulsed noise type waveform in most basic sense. | Pulsed Noise |
| Coherent | Waveform is coherent with the transmit and/or expected receive waveform. Assumed to be closely representing the signal in most basic sense. | Coherent |
| Coherent-Pulse | Waveform is pulse and is coherent with the transmit and/or expected receive waveform. Assumed to be closely representing the pulsed signal in most basic sense. | Coherent |

**Table C-2. Jamming Power Types**

| Jamming Power Type | Description | Mapped Effect Coherency Type(s) |
|---|---|---|
| Noise | Jamming induced power that acts like noise power to a receiver. | None & Non-Coherent |
| Pulsed-Noise | Pulse jamming powered that acts like noise to a receiver. | Non-Coherent Pulse |
| Coherent | Coherent (continuous and/or pulsed) jamming power that acts like a signal to a receiver. | Coherent & Coherent Pulse |

Within most interactions the signal-to-interference (S/I) ratio is calculated using the signal power divided by the noise power + clutter power + jammer power. The jammer power used for the interference is the noise + pulse (non-coherent) jammer power.

### C.4.2 EW Effects Interaction Variables

The following EW effects variable structure is defined for each of the three types of jamming power as well as a separate signal, track and message effect structures. The following tables summarize these two structures and associated variables:

**Table C-3. Jamming Effects Variable Structure**

| Power Effect Variable | Description | Aggregation Type(s) | Default / Undefined Value | Modifying Effect(s) |
|---|---|---|---|---|
| Blanking Factor | Jamming blanking factor (e.g., sidelobe blanker). | Multiplicative | 1.0 | WSF_SLB_EFFECT |
| Cancellation Factor | Jamming cancellation factor (e.g., sidelobe canceller). | Minimum | 1.0 | WSF_SLC_EFFECT, WSF_SLC_DEGRADE_EFFECT |
| Modulation Factor | Jamming processing/modulation type factor, not to physical jamming power factor. | Multiplicative | 1.0 | WSF_POWER_EFFECT |
| Jamming power Factor | Jamming physical power factor. | Multiplicative | 1.0 | WSF_POWER_EFFECT, WSF_COVER_PULSE_EFFECT |
| J/X Factor | Alternate jamming processing/modulation type that has a Jamming-to-Signal/Noise dependency. | Multiplicative | 1.0 | WSF_POWER_EFFECT |
| Target Protection Flag | Flag to specify whether or not jamming power will be allowed to interact with the receiver for a given target or not. | Undefined Boolean | undefined | Base Effect |
| Pulse Suppression Factor | Pulse type jamming suppression factor. | Multiplicative | 1.0 | WSF_PULSE_SUPPRESS_EFFECT |
| Radius Factor | Factor that evaluates the position of the target wrt jammer location to apply a user input factor. | Multiplicative | 1.0 | WSF_RADIUS_EFFECT |

| | | | | |
|---|---|---|---|---|
| Repeater Jamming Factor | Physical jamming power factor dependent upon repeater behavior defined. | Multiplicative | 1.0 | WSF_REPEATER_EFFECT |
| RPJ Factor | Random pulse jamming factor. | Multiplicative | 1.0 | WSF_RPJ_EFFECT |

**Table C-4. Signal Effects Variable Structure**

| Signal Effect Variable | Description | Aggregation Type(s) | Default/Undefined Value | Modifying Effect(s) |
|---|---|---|---|---|
| Signal Power Factor | Signal power factor. | Multiplicative | 1.0 | WSF_POWER_EFFECT |
| Receiver Noise Power Factor | Receiver noise power factor. | Multiplicative | 1.0 | WSF_POWER_EFFECT |

**Table C-5. Track Effects Variable Structure**

| Track Effect Variable | Description | Aggregation Type(s) | Default/Undefined Value | Modifying Effect(s) |
|---|---|---|---|---|
| Azimuth Error | Track azimuth error. | Maximum (EA) / Minimum (EP) | 0.0 | WSF_TRACK_EFFECT |
| Elevation Error | Track elevation error. | Maximum (EA) / Minimum (EP) | 0.0 | WSF_TRACK_EFFECT |
| Range Error | Track range error. | Maximum (EA) / Minimum (EP) | 0.0 | WSF_TRACK_EFFECT |
| Velocity Error | Track velocity error. | Maximum (EA) / Minimum (EP) | 0.0 | WSF_TRACK_EFFECT |
| Track Drop/Maintain Flag | Track drop/maintain flag. | Undefined Boolean | undefined | WSF_TRACK_EFFECT, WSF_SLB_EFFECT (target blanking) |

**Table C-6. Message Effects Variable Structure**

| Track Effect Variable | Description | Aggregation Type(s) | Default/Undefined Value | Modifying Effect(s) |
|---|---|---|---|---|
| Bit Error Rate (BER) | BER for communications device to use. | Maximum (EA) / Minimum (EP) | 0.0 | WSF_COMM_EFFECT |
| Message Drop/Maintain Flag | Message drop/maintain flag. | Undefined Boolean | undefined | WSF_COMM_EFFECT |

### C.4.3 Aggregation Types

The aggregation types given in the table below are used to aggregate (i.e., roll-up) the individual EW effect values into the interaction value to be used by the interaction to apply any EW related effects to the target detection, tracking process and/or message as applicable. All aggregation is done in standard units (i.e., multiplication is the same as addition in dB space.)

**Table C-7. Aggregation Types**

| Aggregation Type | Description |
|---|---|
| Maximum | The maximum of the interaction value and current effect value being applied is taken and used as the interaction value. |
| Minimum | The minimum of the interaction value and current effect value being applied is taken and used as the interaction value. |
| Additive | The addition of the interaction value and current effect value being applied is used as the interaction value. |
| Multiplicative | The multiplied product of the interaction value and current effect value being applied is used as the interaction value. |
| Boolean | A true/false (i.e., two-state) flag that can be toggled based on the current value and logging of the effect. |
| Undefined Boolean | Similar to the Boolean aggregation type, except an undefined state along with the true/false (i.e., three-state) is available as a value. This type can be toggled from undefined (its most common default state) to true/false (i.e., defined) and toggled between the three states thereafter based on the current value and logic of the effect. |

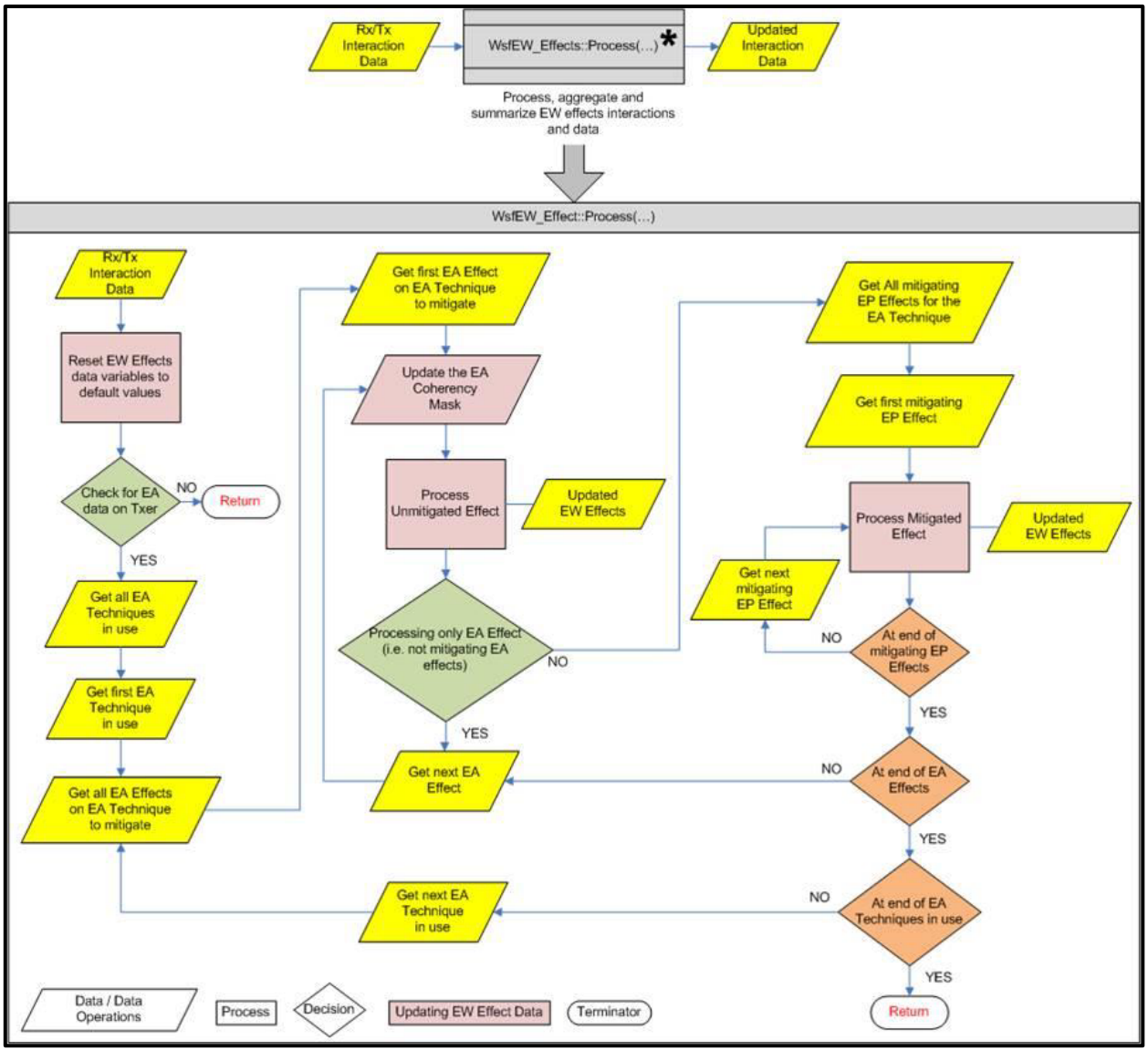


**Figure C-7. AFSIM EW Interaction Flowchart**

# Appendix D- Reactive Integrated Planning aRchitecture (RIPR)

## D.1 Overview

RIPR is the framework included with AFSIM that enables behavior modeling. RIPR is agent based, meaning that each agent acts according to its own knowledge; however, it is common for agents to cooperate and communicate with each other. RIPR is best thought of as a collection of utilities and algorithms that tie together nicely in the construction of an intelligent agent. Most modern RIPR agents, however, do contain a Perception Processor and a Quantum Tasker Processor. The agent senses the world by querying the platform and its subsystems, for information. The agent builds knowledge internally, makes decisions, and then takes action by controlling its platform accordingly. Most platform queries and control actions take place inside of the AFSIM scripting language. The knowledge-building and decision-making actions that RIPR performs are aided by various artificial intelligence technologies summarized here.

## D.2 Perception Processor

A RIPR agent maintains its own perception of threats, assets, and peers. This represents an agent's limited brain and the information can be delayed or erroneous. To represent players of varying skill, each agent has its own tunable cognitive model. For example, an "expert" pilot agent can maintain knowledge of 16 threats that he updates (looks at radar) every 5 seconds.

## D.3 Quantum Tasker

The RIPR Quantum Tasker is used for commander subordinate interaction and task de-confliction. The Quantum Tasker comprises task generator(s), task-asset pair evaluator(s), an allocation algorithm, and various strategy settings (such as how to handle rejected task assignments). Each component (generator, evaluator, allocator) can be selected from pre-defined options, or custom created in script. The RIPR Quantum Tasker tasking system is also compatible with platforms using the older task manager (WSF_TASK_MANAGER and WSF_TASK_PROCESSOR). It can send and/or receive tasks to/from other RIPR agents and other task manager platforms.

The Quantum Tasker's method of operation:

- Acquire perception of assets from cognitive model for matrix columns
- Acquire perception of threats from cognitive model
- Generator generates tasks for matrix rows
- Strategy dictacts how previously assigned tasks, rejected tasks, or new tasks are handled
- Evaluator calculates values for possible asset-task pairs for matrix body
- The allocator runs on the task-asset matrix to find appropriate task allocation, e.g. greedy, optimal, etc.
- Tasks are assigned over comm, handshaking performed for acceptance/rejection

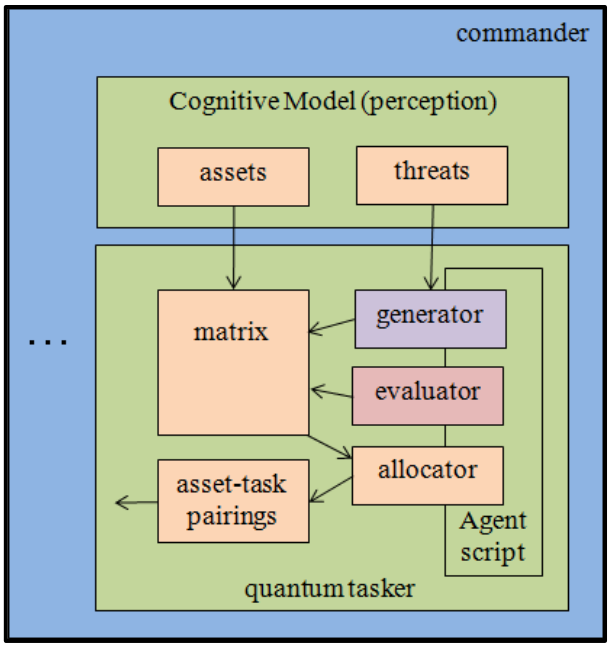


**Figure D- 1: Quantum Tasker Method of Operation**

## D.4 Behavior Tree

RIPR agents typically make use of a RIPR behavior tree to define their behavior. A behavior is a compact modular piece of script that performs some unique action. Behaviors should be parameterized and reusable. A behavior tree allows connection of behaviors in interesting ways so they perform in certain orders or subsets. The whole tree aggregates the behaviors to model an agent's behavior.

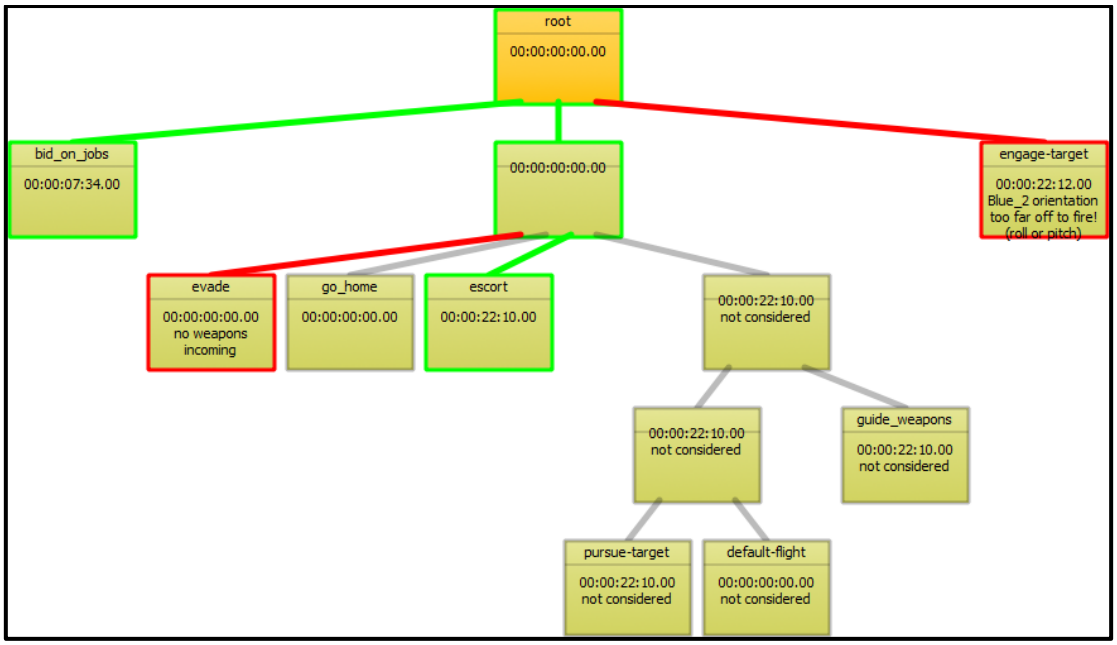


**Figure D- 2: RIPR Behavior Tree**

RIPR behavior trees provide five different intermediate connector-node types:

- Selector – chooses and performs first child behavior to pass its precondition check
- Sequence – performs all child behaviors in sequence until one fails its precondition check.
- Parallel – performs all child behaviors whose precondition check passes.
- Weight Random – makes a weighted random selection from its child behaviors. Weight based on precondition value returned.
- Priority Selector – selects the child behavior who returns the largest precondition value.

Behavior trees provide for maximum utility in developing and editing agents. A properly constructed behavior tree allows a user to find relevant script fast, and swap in other behaviors at appropriate places. For example: try separating out behaviors for choosing desired heading, altitude, and speed from the behavior that actually performs the flight task. When you develop a new flying behavior, e.g. one that used a new route finder, you can swap that for the old one while keeping the logic in place for calculating desired direction.

## D.5 Route Finder

The route finder allows an agent to path around static and/or dynamic obstacles. At this time, obstacles are circular regions defined at a location or attached to another platform. The route finder takes advantage of this assumption to quickly build a search graph around all avoidances and uses a depth-first-search to find the best routes to a target. The route finder has three options for impossible routes:

- Shrink offending avoidances
- Ignore offending avoidances
- Shift start/target points outside of all offending avoidances

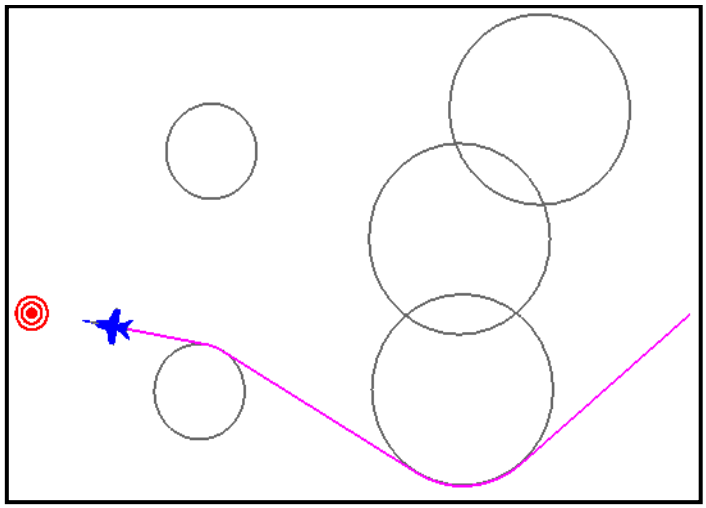

**Figure D- 3: RIPR Route Finder**

## D.6 Heat Map

The heat map allows for the representation of uncertainty in an agent's knowledge of his surroundings. For instance, this capability allows RIPR agents to intelligently attempt to relocate dropped tracks. In reality a when a pilot loses a track he can still have a good idea of where the track was headed, make adjustments based on that intuition and possibly reacquire the track. This is what the heat map attempts to do within AFSIM. It will attempt to figure out where the track has disappeared to, mark these areas hot, check the areas and if the track is not located, it will mark them cold and check other areas marked hot for the track. The hot areas will decay over time if they haven't been checked.

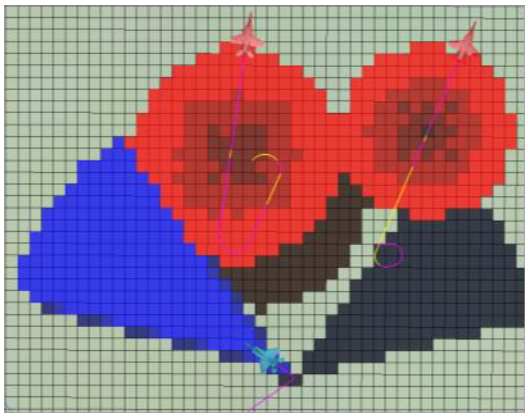

**Figure D-4: Heat Map Example**

## D.7 Cluster Manager

Some RIPR agents take advantage of the Cluster Manager to perform clustering on threat or asset perception in order to think of these larger sets as smaller groups. For example, it is common for a commander to group incoming threats into two clusters so it can send each of its two squadrons after separate groups. The Cluster Manager can cluster based on desired similarity thresholds or based on the desired number of clusters. Similarity measurements can be based on ground distance, 3D distance, or 3D distance and speed. The Cluster Manager can use one of three clustering algorithms:

- Hierarchical Tree Max – default, guaranteed to be optimal, no cluster member dissimilar to any other member past the threshold (this method provides for tighter "classic" groups of members)
- Hierarchical Tree Min – guaranteed to be optimal, no cluster member dissimilar to at least one other member past the threshold (this method allows for long "stringy" chains of members)
- K-Means – not guaranteed to be optimal, fastest, clusters are centered on K different mean points

## D.8 Example agent interaction using all technologies

- A commander agent obtains threats from his cognitive model (Perception Processor)
- Commander's Quantum Tasker generator clusters threats into groups and creates a task for each group

- Commander's Quantum Tasker evaluator scores his squadrons (assets) against each group
- Commander's Quantum Tasker allocator finds optimal task assignment
- Commander assigns task(s) to subordinate flight leads over comm.
- Flight lead uses asset and threat perception from cognitive model while interpreting task
- Flight lead agent's Quantum Tasker generates, evaluates, allocates, and assigns tasks to pilot agents
  Pilot agent uses peer and threat perception from cognitive model
- Pilot agent's behavior tree checks for evade, disengage, bingo conditions
- Pilot agent's behavior tree flies to intercept and eventually engage threat from task
- Pilot agent uses route finder to fly around SAM zones during ingress towards target

## Appendix E - Historical Development of AFSIM

AFSIM is based on The Boeing Company's Analytic Framework for Network-Enabled Systems (AFNES). Under contract, Boeing delivered AFNES to the Air Force (specifically AFRL/RQQD) with unlimited government rights, including source code, in Feb 2013. AFRL/RQQD rebranded AFNES as AFSIM and has started to distribute AFSIM within the Air Force and DoD, including DoD contractors.

The Boeing Company developed and funded the AFNES simulation framework through internal research and development (IR&D) funding from 2003-2014. Beginning in 2005, Boeing began developing a customized AFNES capability to simulate threat Integrated Air Defense Systems (IADS) to assess advanced air vehicle concepts performing Precision Engagement missions. The requirements of this new IADS simulation capability included being able to match results with the Air Force-approved mission level model. The reason for developing an AFNES alternative to the Air Force IADS modeling capability relates to the limitations associated with the Air Force mission level model. Examples of areas in which the Air Force mission level model is lacking include: expansion of representations of Electronic Warfare (EW) techniques; the integration of independent tracking and correlation systems; utilization of vendor-supplied auto-routers and mission optimization capabilities; net-centric communications systems; the contribution of Space assets; and integration of special, existing models, such as AGI's System Tool Kit (STK).

The AFNES IADS capability became operational in 2008, and is currently being utilized by multiple Boeing development programs, as well as government contracted programs, to assess the ability of advanced air vehicle design concepts to penetrate advanced Air Defense networks and conduct precision engagement missions. Because the Air Force is also interested in analyzing future vehicles in the area of persistent and responsive precision engagement, this capability has been briefed to Air Force personnel at various times over the previous five years. In 2010, the AFRL/RQQD Aerospace Vehicles Technology Assessment & Simulation (AVTAS) Lab (formerly AFRL/RBCD) commissioned a trade study of M&S Frameworks for the purpose of assessing potential alternatives to replace or augment their current constructive simulation environment. The result of the AFRL trade study was the selection of AFNES as the best M&S framework to meet their air vehicle mission effectiveness analysis requirements.

Under contract, Boeing delivered AFNES to the Air Force (specifically AFRL/RQQD) with unlimited government rights (including source code) in February 2013. AFRL/RQQD re-branded AFNES as the Advanced Framework for Simulation, Integration, and Modeling (AFSIM) and has started to distribute AFSIM within the Air Force and DoD, including DoD contractors. One of the key focus areas for AFSIM has been the simulated representation of IADS, and the utilization of verified and validated Air Force scenarios and threat representations distributed with the Air Force mission level model. AFSIM provides expanded Modeling & Simulation capabilities to support mission-level analysis studies related to global strike and other research activities.

# List of Acronyms, Abbreviations, and Symbols

| Acronym | Description |
|---|---|
| AFNES | Analytic Framework For Network Enabled Systems |
| AFSIM | Advanced Framework for Simulation, Integration, and Modeling |
| AFRL | Air Force Research Laboratory |
| ANSI | American National Standards Institute |
| API | Application Programming Interface |
| AVTAS | Aerospace Vehicles Technology Assessment & Simulation |
| CC | Conventional Campaign |
| CNR | Clutter to Noise Ratio |
| DEM | Digital Elevation Model |
| DIS | Distributed Interactive Simulation |
| DLL | Dynamic Link Libraries |
| DOE | Design Of Experiments |
| DTED | Digital Terrain Elevation Data |
| EM | Electromagnetic |
| ENU | East, North, Up |
| EA | Electronic Attack |
| EP | Electronic Protect |
| ESM | Electronic Support Measure |
| EW | Electronic Warfare |
| Geo | Geo-referencing |
| GUI | Graphical User Interface |
| HLA | High Level Architecture |
| HTML | HyperText Markup Language |
| IADS | Integrated Air Defense Systems |
| IDE | Integrated Development Environment |
| IR&D | Internal Research and Development |
| ISO | International Organization for Standardization |
| ITU | International Telecommunications Union |
| M&S | Modeling and Simulation |
| MCO | Major Combat Operations |

| Acronym | Description |
|---|---|
| NCO | Network Centric Operations |
| NED | North, East, Down |
| NGA | National Geospatial-Intelligence Agency |
| OSG | Open Scene Graph |
| OO | Object Oriented |
| PIANO | Parametric Integrated Analysis of Objectives |
| RF | Radio Frequency |
| RIPR | Reactive Integrated Planning Architecture |
| RWR | Radar Warning Receiver |
| Wsf Exec | World Simulation Framework Executive |
| SAM | Surface-to-Air Missile |
| SAR | Synthetic Aperture Radar |
| SDB | Scenario Data Base |
| SIMS | Standard Industry Missile Simulator |
| SNR | Signal to Noise Ratio |
| SO | Shared Objects |
| STK | System Tool Kit |
| TDB | Type Data Base |
| TIFF | Tagged Image File Format |
| UCAV | Unmanned Combat Air Vehicle |
| USAF | United States Air Force |
| USGS | United States Geological Survey |
| V&V | Verification & Validation |
| VCD | Vertical Coverage Diagram |
| VESPA | Visual Environment for Scenario Preparation and Analysis |
| VMAP | Vector Map |
| VTK | VESPA ToolKit |
| WGS | World Geodetic System |