Advanced Framework for Simulation, Integration and Modeling (AFSIM)

(Case Number: 88ABW-2015-2258)

Peter D Clive[*], Jeffrey A Johnson[*], Michael J Moss[*], James M Zeh[†], Brian M Birkmire[†], and Douglas D Hodson[‡]
[*]The Boeing Company
St. Louis, MO
[†]Aerospace Systems Directorate
Air Force Research Laboratory
Wright Patterson Air Force Base, OH
[‡]Computer Science and Engineering Department
Air Force Institute of Technology, USA

***Abstract*— The Advanced Framework for Simulation, Integration and Modeling (AFSIM) is an engagement and mission level simulation environment written in C++ originally developed by Boeing and now managed by the Air Force Research Laboratory (AFRL). AFSIM was developed to address analysis capability shortcomings in existing legacy simulation environments as well as to provide an environment built with more modern programming paradigms in mind. AFSIM can simulate missions from subsurface to space and across multiple levels of model fidelity. The AFSIM environment consists of three pieces of software: the framework itself which provides the backbone for defining platforms and interactions, an integrated development environment (IDE) for scenario creation and scripting, and a visualization tool called VESPA. AFSIM also provides a flexible and easy to use agent modeling architecture which utilizes behavior trees and hierarchical tasking called the Reactive Integrated Planning aRchitecture (RIPR). AFSIM is currently ITAR restricted and AFRL only distributes AFSIM within the DoD community. However, work is under way to modify the base architecture facilitating the maintenance of AFSIM versions across multiple levels of releasability.**

***Index Terms*— Simulation Framework, Mission Level Model, Artificial Intelligence Framework, Agent Framework**

I. Introduction

AFSIM is a government-approved C++ simulation framework for use in constructing engagement and mission-level analytic simulations for the Operations Analysis community, as well as virtual experimentation. The primary goal of AFSIM applications is the assessment of new system concepts and designs with advanced capabilities not easily assessed within traditional engagement and mission level simulations. Development activities include modeling weapon kinematics, sensor systems, electronic warfare systems, communication networks, advanced tracking, correlation, and fusion algorithms, and automated tactics and battle management software.

In this section, the reasons for the development and history of AFSIM are presented. The next section provides an overview of the AFSIM architecture, integrated development environment, visualization tools and AFSIM's agent modeling architecture. The following section highlights the current/planned effort to create a Component Based Architecture for AFSIM which will allow multiple levels of releasability. The last section provides a conclusion on AFSIM and its current state.

A. Background

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 rights, including source code, in February 2013. AFRL/RQQD rebranded AFNES as AFSIM and has begun 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. 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.

II. AFSIM Software Suite

The AFSIM software suite consists of three distinct pieces or applications. The first piece is the framework itself which provides the underlying architecture and services allowing the creation of simulation applications. The second piece is the integrated development environment (IDE) which facilitates the creation of scenarios. Lastly the Visualization Environment for Scenario, Preparation and Analysis (VESPA) application allows for post-processing and visualization of scenario executions. This section provides detail on all three.

A. Functional Architecture

AFSIM is an object-oriented, C++ simulation environment that facilitates the prototyping of customized engagement and mission level warfare simulations. AFSIM includes a set of software libraries, shown as a functional architecture in Figure 1, containing routines commonly used to create analytic applications. The AFSIM infrastructure includes routines for the top-level control and management of the simulation; management of time and events within the simulation; management of terrain databases; general purpose math and coordinate transformation utilities; and support of standard simulation interfaces, such as those supporting the Distributed Interactive Simulation (DIS) protocol. The AFSIM component software routines support the definition of entities (platforms) to populate scenarios. These software routines contain models for a variety of user-defined movers, sensors, weapons, processors for defining system behavior and information flow, communications and track management.

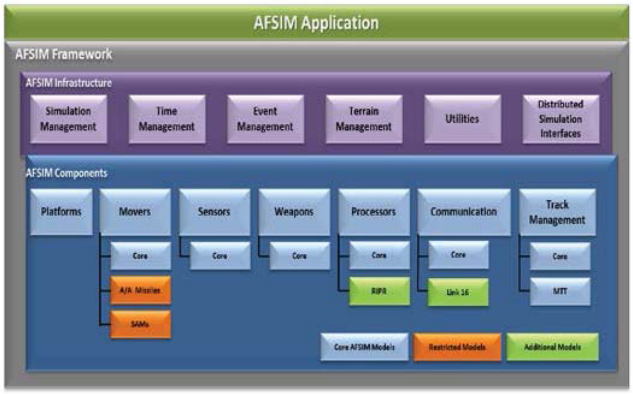

Fig. 1. The AFSIM functional architecture.

The top-level characteristics and capabilities of the AFSIM framework include:

- A class hierarchy of simulation objects, including data driven platforms, movers, sensors, communications networks, processors, weapons, and simulation observers.
- Simulation and Event classes to control time and/or event processing for AFSIM-based models, and the logging of entity data.
- Standard math libraries for coordinate systems (WGS-84, Spherical, ENU, NED), random number generation, DIS communication, High-Level Architecture (HLA) publish and subscribe, and generalized software routines, such as container classes for storing objects and data.
- A common geo-spatial environment and terrain representation, importing standard formats such as National Geospatial-Intelligence Agency (NGA) Digital Terrain Elevation Data (DTED), ESRI, GeoTiff and VMAP database formats.
- A general-purpose scripting language to provide access to framework objects using text input files (i.e., scripts) rather than through the Application Programming Interface (API).
- Communications network modeling, including basic radio transceivers and advanced communications algorithms, including addressable nodes, routers, multi-access protocols, contention and queuing.
- Electronic warfare modeling, including noise and deceptive jamming techniques, as well as the ability to jam and degrade any type of electro-magnetic receiver, including communications systems.
- Modeling of information flow and tasking between player and system elements to define candidate Network Centric Operation (NCO) concepts.
- The ability to run any AFSIM application in both constructive (batch processing) and virtual (real-time) modes.
- User interface elements for integrated scenario generation and post-processor visualization software.

In addition to the AFSIM core, several capabilities are available. Additional capabilities include: multitarget tracking algorithms; Link-16 modeling of both the physical and message layers; and Reactive Integrated Planning aRchitecture (RIPR) intelligent agent algorithms for implementing complex object behaviors. RIPR utilizes a Boeing-developed Quantum Tasker concept for commander subordinate interaction and task de-confliction. Section 3 provides additional details of the RIPR model. Restricted capabilities include missile flyout models.

The baseline AFSIM constructive application is called the Simulation of Autonomously Generated Entities (SAGE), which was one of the first constructive applications developed using the AFSIM framework. SAGE is a simple application that reads in a user-defined input file, executes the simulation, and outputs any user-defined data files. The original purpose

for SAGE was to simulate background air, road or maritime traffic. Although SAGE retains the capability to generate background traffic, the user can exercise all of the resident AFSIM capabilities.

B. AFSIM IDE

AFSIM permits the user to create subsystem definitions in separate files and to include those definitions in a hierarchal manner to define representations. This enables subsystem configuration control and reuse. This flexibility leads to large numbers of subsystem definition files when creating scenarios with a wide variety of different complex systems. The VESPA application facilitates the creation of the scenario initial conditions files. It does not, however, address the problems associated with defining and integrating system and subsystem models or defining system-level relationships such as command chains and peers using ASCII data files. Any input file errors are not discovered until an AFSIM application is executed.

In early 2011, Boeing initiated the development of the AFSIM Integrated Development Environment (IDE) to support the analyst in defining and integrating system and subsystem models. 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 the AFSIM-based application, and visualize the output results and any error messages. This iterative process allows the analyst to receive immediate feedback as system and subsystem models are defined and scenarios are created.

Current capabilities of the IDE support input file creation including support for syntax highlighting, auto-completion, context-sensitive command documentation and a variety of scenario browsers. 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 IDE capabilities are available to assist the analyst in defining system and subsystem models and scenarios.

The 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.

C. Visual Environment for Scenario Preparation and Analysis (VESPA)

To support the analyst, Boeing developed tools to facilitate scenario generation and post-process data analysis and visualization. Specifically, the Visual Environment for Scenario Preparation and Analysis (VESPA) software application was developed to support the 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.

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 2.

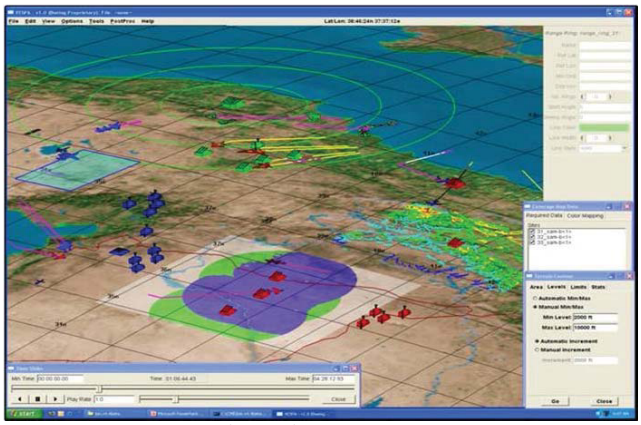

Fig. 2. The VESPA GUI.

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.

III. Reactive Integrated Planning aRchitecture (RIPR)

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 are used to construct intelligent agents. 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 described in this section.

A. Cognitive Model

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. Much of the cognitive model's ability is contained within the Perception Processor.

B. 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. Figure 3 illustrates the various pieces of the Quantum Tasker and their connections.

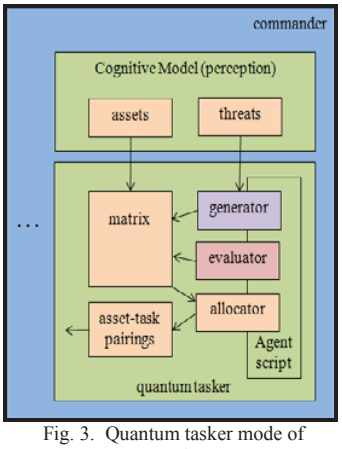

Fig. 3. Quantum tasker mode of operation.

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 dictates 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.

C. 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 4 provides an example of a 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.
- Priority Selector - selects the child behavior who returns the largest precondition value.

Behavior trees provide for maximum utility for 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.

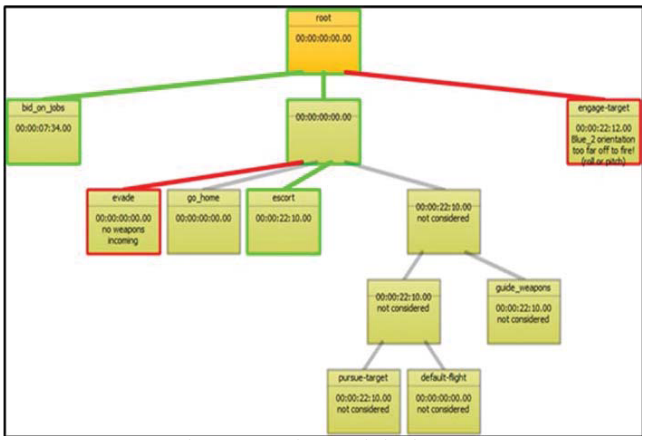

Fig. 4. Example RIPR behavior tree.

D. 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.

E. Example Agent Interaction

Below is an example sequence of interactions within the RIPR architecture for a group of agents:

1. A commander agent obtains threats from his cognitive model (Perception Processor).
2. Commander's Quantum Tasker generator clusters threats into groups and creates a task for each group.
3. Commander's Quantum Tasker evaluator scores his squadrons (assets) against each group.
4. Commander's Quantum Tasker allocator finds optimal task assignment.
5. Commander assigns task(s) to subordinate flight leads over comm.
6. Flight lead uses asset and threat perception from cognitive model while interpreting task.
7. Flight lead agent's Quantum Tasker generates, evaluates, allocates, and assigns tasks to pilot agents.
8. Pilot agent uses peer and threat perception from cognitive model.
9. Pilot agent's behavior tree checks for evade, disengage, bingo conditions.
10. Pilot agent's behavior tree flies to intercept and eventually engages threat from task.
11. Pilot agent uses route finder to fly around SAM zones during ingress towards target.

IV. Future Work

The current state of the AFSIM framework only allows distribution to DoD agencies and DoD contractor's due to International Traffic in Arms Regulations (ITAR) restrictions. It is the desire of the AFRL to allow wider dissemination of the framework in order to provide more modeling and simulation collaboration opportunities. However, the current architecture of AFSIM does not easily lend itself to maintaining multiple versions across multiple release restrictions, which is why an architecture rework is underway to create a Component Based Architecture.

A. Component Based Architecture

Figure 5 details the current base level architecture of AFSIM. Since the base components of AFSIM are directly named in code this makes it difficult to add or remove base component types. Also it is currently difficult to extend other non-platform components.

Fig. 5. Existing AFSIM base level architecture.

In order to better facilitate the ability to add and remove base components work is underway to create a Component Based Architecture, which relies on an underlying generic component class where all components can be derived from. This architecture allows access via naming for components that already exist and will ease the addition and removal of certain component types. This solution maximizes commonality with the original architecture while at the same time providing a means to maintain a release version with no weapons or electronic warfare capabilities included as well as an ITAR release, which would include those components. The new architecture is shown in Figure 6.

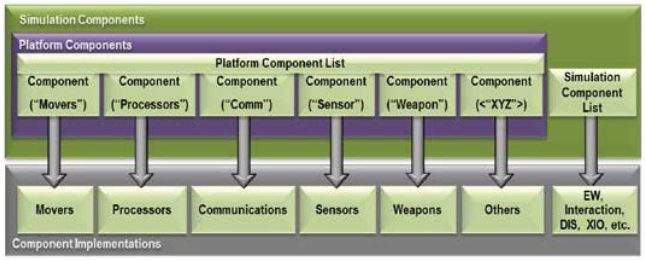

Fig. 6. New AFSIM Component Based Architecture.

V. Conclusion

In this paper we have provided a high level overview of the AFSIM simulation environment. AFSIM has been under development by Boeing under IR&D funds for more than 10 years. Under contract, Boeing delivered AFSIM to the Air Force (specifically AFRL/RQQD) with unlimited government rights (including source code) in February 2013. AFRL has now begun to distribute AFSIM within the DoD community. The AFSIM distribution comes with three pieces of software: the framework itself, an IDE and the visualization tool VESPA. Although AFSIM is currently ITAR restricted future work is planned to modify the underlying architecture to facilitate maintaining multiple versions with varying releasability. Under AFRL management AFSIM will continue to grow as a valuable modeling and simulation tool.