# Integrating AFSIM as an Internal Predictor

Bryan A. W. Jensen[1], Justin Karneeb[1], Hayley Borck[1], and David W. Aha[2]

[1] Knexus Research Corporation; Springfield VA; USA
[2] Navy Center for Applied Research in Artificial Intelligence;
Naval Research Laboratory (Code 5514); Washington, DC; USA
{*first.last*}@knexusresearch.com | david.aha@nrl.navy.mil

15 August 2014

## 1 Introduction

My Summer 2014 internship consisted of contributing to the Autonomy for Air Combat Missions (ATACM) project at the Knexus Research Corporation, a largely collaborative effort involving the Naval Research Lab (NRL), the Naval Air Systems Command (NAVAIR) and the Air Force Research Lab (AFRL). The ATACM project's main purpose is to create an autonomous Unmanned Aerial Vehicle (UAV) capable of flying alongside a human pilot in BVR (Beyond Visual Range) air combat missions. My job as part of Knexus, subcontracted under NRL, consisted of working on the reasoning systems for the UAV, with the label the Tactical Battle Manager (TBM). NAVAIR and AFRL were responsible for interfacing the TBM with their respective simulators and providing domain specific knowledge in the realm of air combat. My contribution to the ATACM project took the form of a predictor, under the name PEPR (the Planned Expected State Predictor), for use in the goal reasoning components of the TBM.

## 2 ATACM Background

ATACM refers to the overall project spanning NRL, NAVAIR and AFRL. ATACM's main goal is to develop the controller for an autonomous UAV, which has to be capable of flying wingman to a human pilot in BVR combat; this involves creating a system capable of observing the environment as well as receiving input from the human pilot in order to determine an intelligent course of action.

The work I did for the ATACM project built on top of preexisting systems, a few of are briefly described below.

### 2.1 TBM

The TBM, partially developed at Knexus Research, refers to the the suite of functionality to control the UAV's actions. It achieves this for the most part through use of a goal reasoning process (to be described later), utilizing components such as a Behavior Recognizer, planner, negotiator and predictor.

### 2.2 AFSIM

AFSIM (Air Force Simulator, shown in Figure 1) is one of the two simulators involved in the ATACM project for use as a testing environment in which to run the TBM, the other being NGTS (Next Generation Threat System). However, AFSIM was the one chosen for use internally as a prediction mechanism due to its ability to run much faster than real-time, due to a decreased level of fidelity, which allowed for a prediction in less than a second. Throughout our work on the TBM we were in communication with the current developers of AFSIM, working out of AFRL.

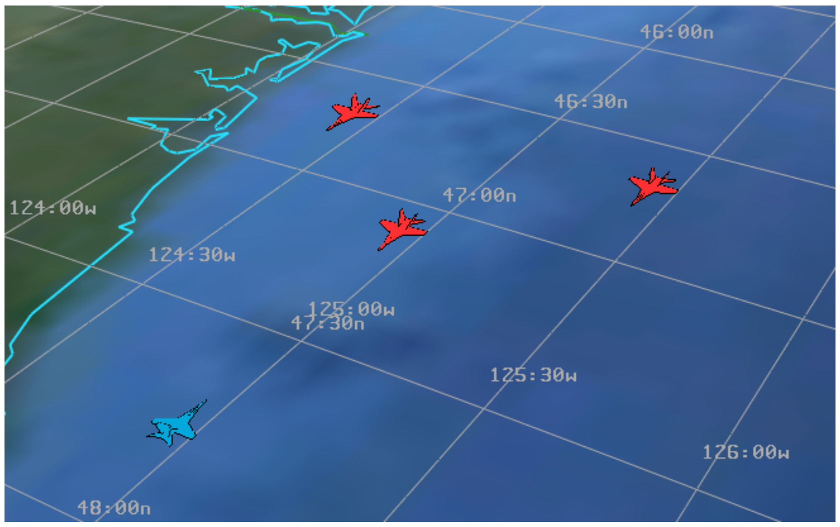

**Fig. 1.** Screenshot of AFSIM in action, with one UAV and three hostiles.

### 2.3 PEPR

PEPR is the name given to the prediction functionality built around the use of AFSIM as an internal simulator. It is on this system that I spent the majority of my time, developing it for use in the TBM architecture.

## 3 My Work

### 3.1 Starting with AFSIM

When I started work at Knexus as a part of the ATACM project, locally there was little knowledge or familiarity with AFSIM as most of the development had been done in interface with NGTS, the simulation platform provided by NAVAIR. Therefore my first task was to learn and understand the internals of AFSIM, as well as what was needed to integrate the simulator into the TBM. This learning process consisted primarily of familiarizing myself with the AFSIM scripting language and grammar, necessary components for constructing a scenario to be simulated. I spent a fair amount of time in the provided IDE altering scenarios to determine their functionality and conceptually how AFSIM would fit into the TBM. At the same time, I was also examining the source code for the executable that accepted these source files and simulated the scenario, in order to learn how to adapt it to our system.

## 3.2 Creating AFSIM's Interface

After determining the process by which AFSIM understands a scenario, represented in script files, the next stage was to create a programmatic interface to replace the existing executable. This would allow us to run AFSIM with dynamically created scenarios, designed to test the feasibility of a given plan.

It was at this stage that I got my first real experience with work on a large code base, none of which was written by anyone with whom I had direct contact. I also gained a lot of experience with topics that I had yet to see used to great effect, such as factory classes and singletons.

The end result of this stage was small wrapper library which we could import into the TBM, replacing the provided executable. This new component allowed us to programmatically provide a string, which represented the script file, and subsequently run the simulator.

## 3.3 The TBM Structure

After the functionality was finalized for the running of dynamic simulations, the next step was to figure out where PEPR fit into the framework laid out for the UAV's reasoning system. The TBM encompasses the entire reasoning process and decision making of the UAV, and is partially depicted in part in Figure 2. The main component of the TBM, depicted in green in the center of the diagram, is the goal reasoning process, around which the cognitive system was built. Along the left is the information handling system where events such as radar input are handled and converted into useful data. On the right hand side can be seen the planning portion of the TBM, with the negotiator acting as a go-between for the planner and the goal reasoner. At the top are the systems for actual actions taken by the UAV, such as the plan executor and, in the current setup, the simulation wrapper which interfaces with the external simulator in which the entire system runs.

**Goal Reasoning** The primary driving force behind the cognitive system of the TBM is the goal reasoner. Goal reasoning is a process in which a set of goals is maintained and, in our system as in the BDI (Belief, Desire, Intention) architecture, a dynamic list of of desires is kept and monitored. The goal reasoner is responsible for choosing the best goal(s) to optimize the values of the desires.

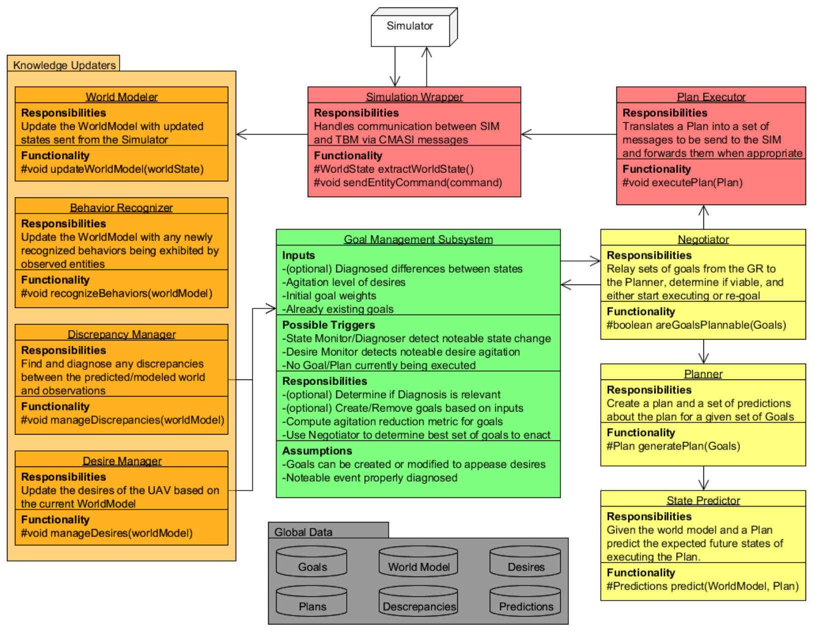
Simulator
Knowledge Updaters
World Modeler
Responsibilities
Update the WorldModel with updated states sent from the Simulator
Functionality
#void updateWorldModel(worldState)
Behavior Recognizer
Responsibilities
Update the WorldModel with any newly recognized behaviors being exhibited by observed entities
Functionality
#void recognizeBehaviors(worldModel)
Discrepancy Manager
Responsibilities
Find and diagnose any discrepancies between the predicted/modeled world and observations
Functionality
#void manageDiscrepancies(worldModel)
Desire Manager
Responsibilities
Update the desires of the UAV based on the current WorldModel
Functionality
#void manageDesires(worldModel)
Simulation Wrapper
Responsibilities
Handles communication between SIM and TBM via CMASI messages
Functionality
#WorldState extractWorldState()
#void sendEntityCommand(command)
Plan Executor
Responsibilities
Translates a Plan into a set of messages to be send to the SIM and forwards them when appropriate
Functionality
#void executePlan(Plan)
Goal Management Subsystem
Inputs
-(optional) Diagnosed differences between states
-Agitation level of desires
-Initial goal weights
-Already existing goals
Possible Triggers
-State Monitor/Diagnoser detect noteable state change
-Desire Monitor detects noteable desire agitation
-No Goal/Plan currently being executed
Responsibilities
-(optional) Determine if Diagnosis is relevant
-(optional) Create/Remove goals based on inputs
-Compute agitation reduction metric for goals
-Use Negotiator to determine best set of goals to enact
Assumptions
-Goals can be created or modified to appease desires
-Noteable event properly diagnosed
Negotiator
Responsibilities
Relay sets of goals from the GR to the Planner, determine if viable, and either start executing or re-goal
Functionality
#boolean areGoalsPlannable(Goals)
Planner
Responsibilities
Create a plan and a set of predictions about the plan for a given set of Goals
Functionality
#Plan generatePlan(Goals)
State Predictor
Responsibilities
Given the world model and a Plan predict the expected future states of executing the Plan.
Functionality
#Predictions predict(WorldModel, Plan)
Global Data
Goals
World Model
Desires
Plans
Descrepancies
Predictions

**Fig. 2.** The Tactical Battle Manager

Goals are a partial state of the world some time in the future; some example goals are "keep hostiles away from this zone," or "destroy target hostile."

Desires represent some metric of a given state, determining how much we "like" that given state in that specific regard. For instance, a desire may be "safety for the UAV," "safety for our wingman," or "complete the mission."

The goal reasoner is responsible for choosing goals that will result in the best status of the desires, and to do so it interacts with the negotiator in order to determine a plan of action to accomplish said goals.

**Negotiator** It is through the negotiator that the goal reasoner makes use of the planner. The goal reasoner may come up with a goal that it believes will best suit its desires, such as "destroy the hostile," "dont́ get destroyed ourselves," and "prevent the base from being attacked." The negotiator then requests that the planner generate a plan that will complete these goals. Should the planner not be able to do so, the negotiator is responsible for informing the goal reasoner about the infeasibility of a given goal set, after which the goal reasoner will generate a new set and try again.

**Planner** The planner is in charge of turning a high level, conceptual plan into a sequence of executable events. Our planner currently relies heavily on predictions to create a plan, utilizing the AFSIM simulator to do so. It is within the planner that the prediction system PEPR is implemented.

### 3.4 StatePredictor

At this point in the development we replaced the existing, rudimentary negotiator/planner combination with one based on behaviors and prediction. "Behaviors" is a term with various definitions, but we use it to mean the high level description of the actions of the platform, such as "aggressive," "passive," or "defensive". As a first draft of the system, and in order to quickly produce a functioning product, we decided on a planner/predictor setup that relied heavily on behaviors in order to dictate the actions of not just hostiles and allies but the UAV itself. It was in this system that PEPR was to be implemented and integrated into the existing TBM architecture.

**Integration** At this point I moved from working with AFSIM to the TBM. For me, AFSIM was a large code base and entirely foreign to both me and my co-workers, while the TBM was one that had been built from the ground-up by the technical lead in the past months. I started working on integration of the wrapper around AFSIM, a component written in C++, with the intent of making the functionality available to the TBM, a project written entirely in Java. This language mis-match ended up being a sizable hurdle, with the eventual solution being dynamic linking of the custom C++ AFSIM library into the Java code through the JNI library.

**AFSIM Wrapper** Once we had a method of calling AFSIM from within the StatePredictor, all that remained was to create a suite of functionality to provide conversions between the data format from the TBM and the formats that AFSIM requires for its script files. This stage of development represents the majority of the work done on the PEPR sub-project, and required some interesting concept intersections where similar ideas were handled differently in the TBM, a system primarily familiar with the NGTS concepts, and AFSIM; both covered very similar concepts, but often just slightly differently.

For instance, AFSIM defaults to the input method of positions via LLA (latitude, longitude, altitude), in the Degrees:Minutes:Seconds format. Internally, the TBM stores all positional data as Earth Centered, Earth Fixed (ECEF) coordinates, which is a more typical Cartesian system of (x, y, z) data points. The TBM contains all knowledge about the UAV as well as allied and hostile aircraft as a concept of "Models," containing a history of states with information such as position; AFSIM accepts a component called a "platform," which contains data about its capabilities and components, e.g. radar, weapons, movement, etc.

Conceptual conversions were not the only type performed between AFSIM and the rest of the TBM. As AFSIM's main form of output is a text file containing "Events," each being space-separated data in what amounts to a string, we had to build a robust system to parse this information into a future set of states to store in the "Models" for the TBM.

These are but some examples of the work that PEPR has to perform to interface the simulation and prediction capabilities of AFSIM, in order to make the tool usable as part of the TBM architecture.

## 4 Conclusion

This summer's work has predominantly provided me with an opportunity to gain experience in a multitude of ways. I've gained experience with a working environment, where I was handling much the same duties as a typical employee and working alongside others to accomplish a project. And I've also gained experience with areas that aren't necessarily possible in an educational environment, such as working with a foreign, evolving code base, i.e. AFSIM. Over two months of writing code for a sizable project, I learned a lot in terms of applications of concepts that had previously been touched upon, but not presented themselves as especially useful. Nine weeks of realistic, hands-on experience on a project with a defined goal, deadlines, expectations, needs, uses and purpose have been invaluable to me as a relatively new programmer, and I am very grateful that I was able to get such an opportunity.

## 5 Acknowledgments

First of all I wish to express my thanks to the Naval Research Lab for providing the funding for myself and the ATACM project as a whole. I also want to thank Justin Karneeb, the technical lead for ATACM at Knexus, for being my unofficial mentor, and to Hayley Borck for filling a similar role. Most of all, however, I am very grateful to David Aha for giving me the opportunity to work on the ATACM project as well as for filling the role of the project head.