Proceedings of the 2007 Winter Simulation Conference
S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, eds.

SELF ORGANIZED UAV SWARM PLANNING OPTIMIZATION FOR SEARCH AND DESTROY USING SWARMFARE SIMULATION

Dustin J. Nowak
Ian Price
Gary B. Lamont

Department of Electrical and Computer Engineering
Air Force Institute of Technology
2950 Hobson Way Bldg 640
WPAFB, Dayton, OH 45433, U.S.A.

ABSTRACT

As military interest continues to grow for Unmanned Aerial Vehicle (UAV) capabilities, the Air Force is exploring UAV autonomous control, mission planning and optimization techniques. The SWARMFARE simulation system allows for Evolutionary Algorithm computations of swarm based UAV Self Organization (SO). Through Swarmfare, the capability exists to evaluate guiding behaviors that allow autonomous control via independent agent interaction with its environment. Current results show that through an implementation of ten basic rules the swarm forms and moves about a space with reasonable success. The next step is to focus on optimization of the formation, traversal of the search space and attack. In this paper we cover the capabilities, initial research results, and way ahead for this simulation. Overall the SWARMFARE tool has established a sandbox in which it is possible to optimize these and build new behaviors.

1 INTRODUCTION

The on going struggle in military doctrine is how to reduce the kill-chain (the time between target acquisition and employment of weapons) and maximize force multipliers (putting as few personnel in harms way while still accomplishing the task). As a result the new wave of UAV research is focused on the increasing need for Autonomous, Self Organized UAV platforms for reconnaissance and delivery of munitions (conventional and non-conventional).

In this pursuit the Air Force Research Laboratories (AFRL) and Air Force Institute of Technology (AFIT) have developed two systems to research and develop the software for these systems. The AFRL tool, Multi-UAV is MATLAB based, that is focused heavily on the mathematics and kinematics of the flight and communications of UAVs in a swarm. SWARMFARE, originally developed by Ian Price (2006), is focused on architecture and algorithms to create functioning autonomous UAVs. Although both systems are very capable, they each have a different focus. The MATLAB tool is more mature through production level design and software engineering.

The SWARMFARE system has a lot of potential because it's a Self Organization (SO) sandbox in which many varying scenarios and behaviors can be tested. Its viewer allows for close analysis of the system's results. There are several functions that allow for the analysis of the backbone Genetic Algorithm (GA) data, as well. With a robust library and a good interface the system is an extremely powerful analytical tool.

This paper describes the SO background and how it is implemented. The current state of the Swarmfare simulation is discussed along with recent improvements. And, finally the future of this research and the SWARMFARE system is addressed.

2 BACKGROUND

The original impetus for building the foundation of SWARMFARE was to establish a solid SO sandbox system while creating a simulation that could still reasonably emulate UAV kinematics and communications. Price developed this system as distributed java system, platform independent, and using a Genetic Algorithm developed a balanced SO structure.

2.1 Self Organization Characteristics

Studies of Self Organization (SO) in natural and biological systems abound (Camazine et al. 2003, Heylighen 2006). The crux of SO rest on three points: the lack of global knowledge for any one agent, a simple rule set defining the behaviors, which results in emergent properties

that are greater than the effects of any single agent. Many examples of this phenomena exist in nature: from insects (bees and ants) to mammal herds and avian flocks. In none of these instances does any one agent (animal) have global knowledge of the system or its environment. Agent interaction is localized and decisions are based on a simple but clear set of rules. Through these actions emergent properties and structures such as ant mounds and paths (Bonabeau et al. 1999) to flocking to synchronize flashing of fireflies (Camazine et al. 2003) are formed.

There are many advantages to this type of system in the computational world: dynamic adaptation, scalability, and redundancy (Cottam, Ranson, and Vounckx 2003). These advantages would be limited by the creator's design if structure is implemented using a top-down approach. Although current SO systems are still not expansively used or mature, they are showing comparable effectiveness to the top-down systems they are intended to replace. They also provide the added advantage of self initiated growth through the aforementioned properties.

2.2 SO System Modeling

In modeling SO systems, there are several requirements that must be met. The swarm behavior must emerge, no direct control. With this the real design happens in the low level behaviors. Also the control must be completely autonomous. As in most cases, there are multiple modeling approaches to structuring the SO systems.

Gat (1998) uses an approach focused on the most difficult aspect of autonomous control, stimuli response. In order to accomplish quick response in a dynamic environment, the system breaks the structure into three parts. The *Controller* focuses solely on immediate response with little state knowledge. The *deliberator* does all of the deliberate planning and the third communicates and coordinates between the first two, the *Sequencer*. The current implementation of this Swarmfare the environment is static and therefore is not necessarily concerned with the instantaneous response.

Rubio, Vagners, and Rydsyk (2004) use a model that focuses on the level of knowledge: *world states, local states, vehicle states.* Through this structure, he implemented autonomous maritime vehicle control.

Reynolds (2005) also implements a structure when trying to establish control for autonomous agents. In his "game" hierarchy, he establishes a three level structure: *Action Selection*, *Steering*, and *Locomotion*. This forms the fundamental structure for his game agents to navigate game space. This model provided the initial foundation of our investigation and development.

Rosenblatt (1997) proposes a system that again separates the vehicle control from the learning, this time not so directly focused on the instantaneous response. His model has a set of modes that manages weight values for several control modules in the *arbiter*. The values from the *mode manager* and the modules are used to calculate the next step and are then sent to the agent *controller*.

SWARMFARE SO modeling combines the described structures into a three tier structure similar to that illustrated in Figure 1. Here the simulation revolves around the development of SO rules and the interplay of those rules. This takes away from the criticality of the immediate response ability. In the simulation, the visualization represents the states of the swarm as the world, UAVs locally and the rules as locomotion. That is mapped high to low as System State (the Flock), UAV Agent State (BAs), and Update Local State through transition functions (SO rules). The first level maps the real world system to the set of knowledge and operators, this layer is called the SO System modeling. The second, transitory layer, consist of two codependent sub-layers. Translating the SO system model mapping into computational terms is the first sub-layer and the structure of the code to be implemented from the above design is the second. Finally this design is mapped to code in the implementation layer.

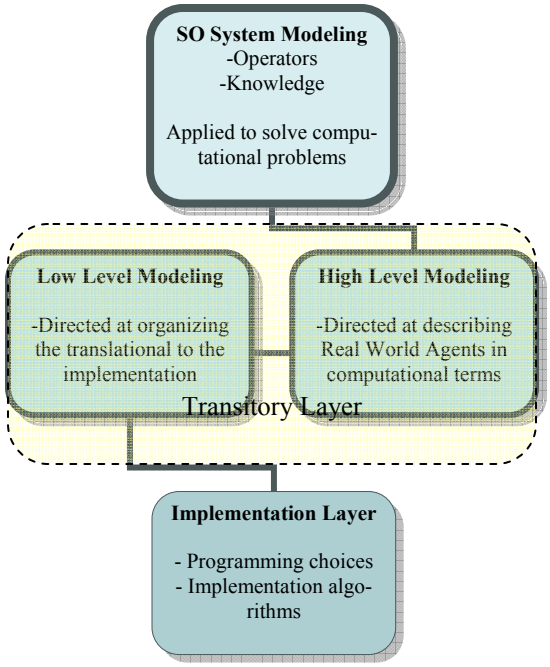

Figure 1: Three-tier SO architecture.

Top-level SO model. Although Price (2006) used the term "High Level Design" to mark the beginning of his mapping from real-world SO to the computational world, the top level is still present. In the case of Swarmfare the SO system chosen to model most closely resembles a flock of birds. The structure evolved from Reynolds (1987). The SO model provides a knowledge base and many operators, including inter-agent detection, the three function of flocking, cohesion, collision avoidance and vector matching. On top of this, Price includes goal seek-

ing, sensor interpretation, pheromone based attack operators, and unidirectional interagent communication.

Behavior Archetypes. When SO is applied to a system rules must be drawn together in order for an agent or computer to accomplish a coherent response. In the case of SWARMFARE, the system gathers these rules together to form Behavior Archetypes (BA). Through these groupings the rules are weighted and applied to establish each subsequent action. This is similar to the modes in Rosenblatt's architecture. Through this open scheme of BAs, the system could theoretically be loaded with dozens of different schemas that would optimize the reaction to a situation at any given point. For instance currently there exists a BA focused on the rudimentary accomplishment of searches. This BA allows for the maintenance of looser group cohesion in order to maximize the swarm's collective sensor footprint.

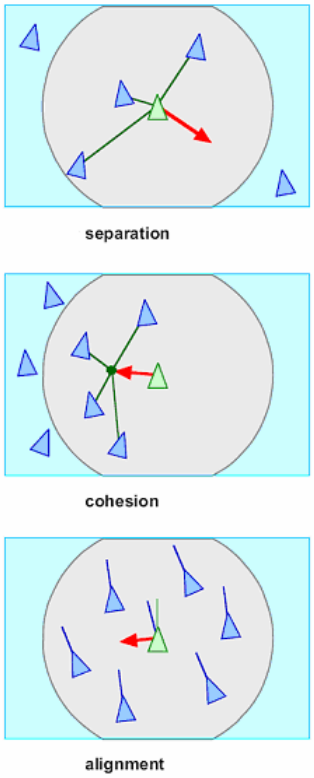

Figure 2: Reynolds' (1987) three fundamental swarming rules.

Rules Utilized. Thus far the SWARMFARE simulation combines 10 rules, as articulated below, to define the BAs:

- *Flat Align – vector align with neighbors*
- *Target Orbit – orbit target at safe distance*
- *Cluster range towards - cohesion*
- *Cluster Range away - separation*
- *Attract – towards center of mass of all targets*
- *Weighted Attract – towards closest target*
- *Target Repel – repel if with 90% of UAV sensor range*
- *Weighted Target Repel – repulsion based on proximity to target*
- *Evade – apriori collision detection and avoidance*
- *Obstacle Avoidance*

The ten rules are derived from the two behavior archetypes, swarm and target interaction. The swarm rules, flat align, separation, cohesion, obstacle avoidance and evade some from Reynolds (1987, 1999). The target driven rules, attract (and weighted), repel (and weighted), and orbit are derived. Attract and repel tries to form a balance of aggression and respect towards targets. The orbit stems from Lua (2003) where each one is weighted differently depending on the makeup defined by Behavior Archetypes of the agent.

Genetic Algorithms. In order to truly develop rules that apply effectively in an SO system, the parameters and responses must be developed bottom-up. SWARMFARE exploits Genetic Algorithms (GA) to accomplish this feat.

Through a set of chromosomes, which map weightings of the SO rules to integers, a standard GA is used to grow successful chromosome. Although there are multiple behaviors and rules, the algorithm collects them all into one fitness function value. The fitness function is then used in elitist selection of the surviving chromosomes. These rule weights in the chromosome are then combined with the BAs to define the next move, again similar to the arbiter in Rosenblatt (1997). After about 50 generations of using this heuristic, the system not only operates effectively as a swarm but also has developed the ability to react to various situations outside of its trained domain.

Parameters. In genetic algorithms the system uses multiple inputs to define the method in which the system allows the chromosomes to develop. In SWARMFARE the controllable parameters include: mutation rate, crossover rate, random and non-random chromosome production levels, and survival rates. Mutation is simple a percentage chance of random mutation. Genetic crossover uses a similar probability to control it. The random population number defines how many completely random, not resulting from crossover/mutation, chromosomes are created. Non-random population defines the number resulting from the crossover and mutation steps. Finally the number that survives a generation controls the overall population size.

3 SWARMFARE SIMULATION CAPABILITIES

To better understand the use of Swarmfare as a simulation for research on UAV Swarms the capabilities and flexibility of the systems models must be illustrated.

3.1 Agents

The fundamental building block of this system is the agent. Its simple rule sets give the agents robust capabilities. Agent knowledge includes its location, its flight kinematics (modeled loosely after the Predator) (UAVForum), hit points (used for basic life capabilities in engagements), attack rate, and sensors. The operators include inter-agent communication, formation flight, search and attack. There are several sub attributes that make these operators function properly. The ten aforementioned rules facilitate these properties through emergence.

The original investigation using this system established the capability to use a homogeneous or heterogeneous group of UAVs. This paved the way for dynamic rule sets which depend on the swarm makeup and attributes. As part of heterogeneity, different UAVs are capable of different types of sensors and ranges. This assumes that the sensors interpolate the images during the target acquisition process, and the only concern to the system is the distance at which a target can be accurately acquired. This flexibility allows the rules sets to be applied generically across any swarm composition.

Another operator of the agent is its ability to accomplish implicit or explicit communications with the other agents or to accomplish no communication at all. Testing on this parameter can show the usefulness of different communication setups.

3.2 Communication

Communication, as most people think of it in the real world, happens through explicit commands or dialogue from agent to agent. Implicit communication, stigmergy, which is usually forsaken in discussions about the subject, is extremely powerful in the SO world. For example, an ant communicates, implicitly, to another ant when building through the placement of material (Camazine et al. 2003).

In the SWARMFARE simulation, the vast majority of communication is implicit. There are two pieces of information being transferred: a UAVs current position and vector. This information is only transmitted to adjacent UAVs. At no point, however, do neighboring agents communicate commands such as “Turn left or right”. A great example of this is when the swarm finds a target. The first agent senses the target, communicates the target position, then the rules turns the acquiring UAV towards the target. Through implicit communication the whole swarm moves to follow the first agent maintaining the swarm formation.

SWARMFARE use pseudo communication to transmit all of its interagent messages. Because all of the UAVs are in the same simulation, there is no communication transmission medium. The simulation publishes the individual UAVs information to a data blackboard that limits access according to the UAV’s relative location in the domain space. This modeling of communication could be improved to more accurately represent the time and constraints of wireless data networks. The task of the mobile interagent communications are beyond the abstraction needed to implement and develop SO swarms and the best behavior rule sets.

3.3 Environment

SWARMFARE is capable of creating several different types of environmental pieces that can be combined to create a realistic problem domain space. There are two fundamental pieces. The agents construct is used for UAVs and also used as aggressors/targets. The other type environmental pieces are obstacle lines, which are used for obstacles of all forms and the boundaries to the simulation area.

Targets. In this system targets are no more than immobile agents with a separate set of markers to allow for distinction of friend and foe. Although they are currently implemented as stationary, the foundation exists for mobile targets. As with the inherited mobility capabilities, inter-target communication is a set of operators that can be chosen to be applied or not.

Radars. Simulation of a radar site is allowed. This is accomplished by use of a target. It is set up with a large sensor radius, no aggression capability, and communication is allowed to other aggressors.

Obstacles. This simulation is also capable of obstacle avoidance. The parameters for these obstacles must be articulated in the scenario file. Note that the agents are capable of accomplishing obstacle avoidance.

Simulation Domain. Constraining the domain space of the swarm focuses the search techniques on a digestible level of information. Through abstraction, the rules formed in a geometrical structure (box) could be applied in large environments. This is simply implemented as obstacle lines to create an outer box in which the UAVs avoid these lines in a similar fashion as the original obstacles.

3.4 SwarmForge User Interface

The SwarmForge interface allows for easy control of the all the capabilities of the SWARMFARE system. There are essentially three parts to the Graphical User Interfaces (GUI), Chromosome development, chromosome testing

and visualization. The Server, Client and RMI tabs shown in Figure 2 control the GA farming model used to develop the chromosomes. This interface allows for distributed computing. In section 3.6 the implementation of a parallelized version is discussed. Here all of the inputs to control the Genetic algorithm are accessible. Everything from the numbers in the population to the number of generations can be controlled. As well several basic simulation variables such as probabilistic path following and active communications can be manipulated.

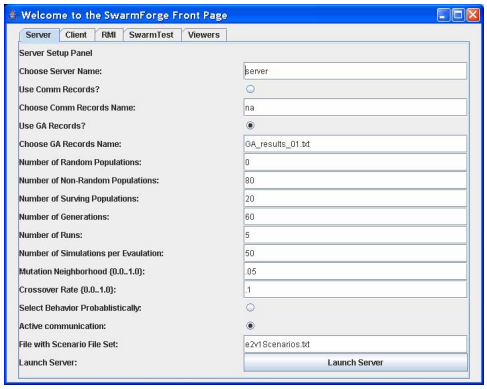

Figure 3: Swarmforge GUI open to the server tab.

The testing section which is shown in Figure 3 controls the testing of a resulting chromosome from the development stage. The resulting chromosomes can be tested against any scenario with the same attributes. In this case it is suggested that homogenous testing scenarios only utilize homogeneous chromosomes and the same for heterogeneous, specifically assuring similar swarm compositions.

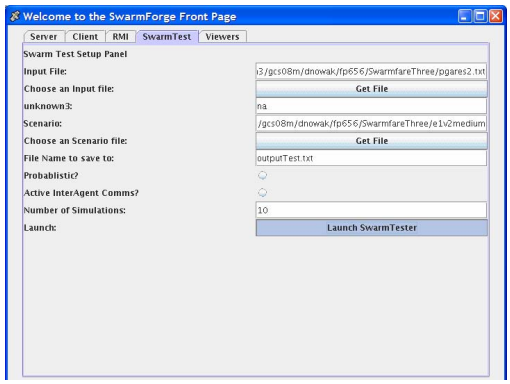

Figure 4: Swarmforge GUI open to the testing tab.

After the testing phase completes, the system can run a series of visualizations. The visualization includes script view, path view and field view. The first allows for visualization of the agents as they dynamically move through the environment. The second shows the path of every UAV in the swarm through the entire scenario. The field view allows for visualization of the vector fields defined be the rule sets.

3.5 Visualization

The need for the visualization stems from the ability to comprehend the driving forces behind the SO behaviors developed. Trying to interpret the resulting chromosomes that define the behavior sets is quite difficult. The weightings simply imply strengths in the resulting vector of that rule, which is then aggregated into the agent's new vector. With this visualization the result of those weightings and vector aggregation can lead to understand and validation that the system developed successful behavior.

SWARMFARE has several visualization tools. The main visualization package Swarmfare Viewer takes scripts developed during the evolution of the chromosomes on a specific scenario and plays them out in a temporal manner. This enables easy digestion of the swarm capabilities of that chromosome set. In this visualization, UAVs, targets, and obstacles are all displayed with various attributes, such as sensor range rings, which are available for display as well. In this viewer it is possible to select a target or UAV and follow the movements of that agent.

There is another viewer that is capable of giving the path plot information in a non-temporal state. All of the paths followed by the UAVs are drawn throughout the landscape. In this visualization there is no need for additional data displays or flight following.

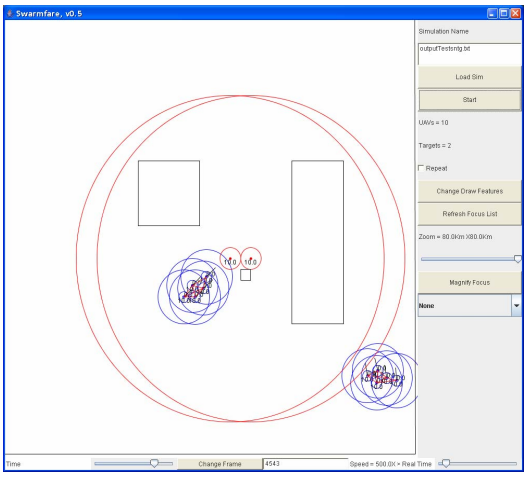

Figure 5: Swarmfare script visualization.

Color The use of color in visualization allows the eye to quickly interpret data. In these two modes there are several color schemes. First the red points and rings define the targets, with the outer ring representing detection and the inner engagement range. The obstacles are defined by a non-threatening black. The UAVs use several color sets. Blue defines their location and detection and

threat rings, similar to the targets. The communications links between the UAVs are shown as well. Red lines indicate active connections between UAVs, when the link turns green one of the UAVs has acquired a target. The line at the front of the UAV indicates the instantaneous vector heading.

For the purpose of understanding the system and trouble shooting during development there also exists three other visualization tools. The Field Plot Viewer, Sensor Value Viewer and Behavior Set Viewer are all additional tools in the system. The field plot displays any pheromone or attraction gradient around targets and UAVs. The Sensor Values Viewer shows a UAV's particular sensor capabilities in a given situation. The Behavior Set Viewer shows the gradient vectors for any of the given behaviors.

3.6 Parallelization

We have implemented a panmictic population parallelization is also known as Master-Slave (MS). Here the system parallelizes the fitness function with the effect of reducing overall computational time. In this form there are no effects on the results of the GA. Communication happens when a chromosome or set of chromosomes receives its evaluation and returns to the population. The processing of the fitness function is then divided by the number of processors available as slaves. In Equation (1), the resulting parallel time calculation which is:

$$T_p = k(t_s + t_w m)(p\text{-}1) + \frac{n^{ck}}{p}. \quad (1)$$

In this equation k is the number of generations, t_s and t_w are the setup and transfer time for one word of communications, p is the number of processors, n is the number in the population and c is a constant defining the variable. Note the linear complexity of the fitness function. In this n^{ck} represents the linear calculation time dominated by the fitness function.

Testing The system was run through a series of tests. This series was geared towards establishing the ratio of chromosomes to the processor population. To define this ratio, population sizes of 5, 15, 25, 45, and 65, were run on 1, 4, 8 and 16 processors. Each trial had the parameters of 5 generations with 5 runs per generation performing 5 simulations per fitness evaluation.

Figure 5 shows the time results from the Farming implementation. We varied the process sizes as well. Note variance is shown but through thorough testing the system reflected relatively stable results.

In Figures 6 and 7, the results testing various population sizes is shown. Empirically the system seems to indicate that the best population for 16 processors is between 45 and 65. Those two rates reflect the highest speedup at that level. However a population of 25 also rated closely making it difficult to winnow in on the exact ratio with this data. Trends definitely indicate around 2 to 3 populations per node, in agreement with the mathematical analysis. The equation defining the loading comes from Cantu-Paz (2005):

$$P^* = \sqrt{\frac{nT_f}{T_c}}. \quad (2)$$

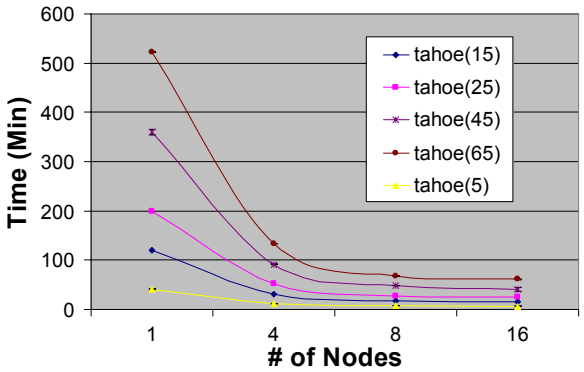

Figure 6: Farming process time (varying population, 5 – generations, runs and simulations).

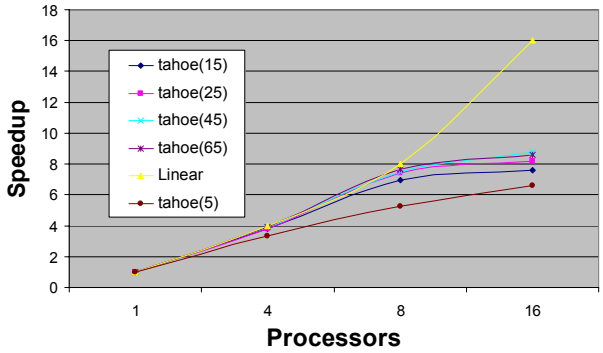

Figure 7: Farming speedup (varying population, 5 – generations, runs and simulations).

3.7 Analytical Tools

While running the tool that grows the chromosomes, the system can output large amounts of data about each generation, simulation, and run. All of this information is transferred directly to spreadsheets for analysis. The data collected includes each runs fitness function score, the sensing values of the swarm, the target discovery percentages and the target destruction percentages. This data can be consolidated to establish trends and validate the growth of more refined chromosomes.

4 SIMULATION DIRECTION

In this section we outline some of the areas in which the system can be improved. None of these improvements effect the verification of the system but would add to the validity of its real world application.

4.1 JAUS Architecture

Department of Defense (DOD) has published the Joint Architecture for Unmanned Systems. (OUSD 2007) The purpose of this publication is focused on giving the government non-proprietary systems that integrate well into existing infrastructure and networks. SWARMFARE is compliant with the intent of the document; however, additional work is required. The documentation and configuration of the system does not currently exist in the JAUS format.

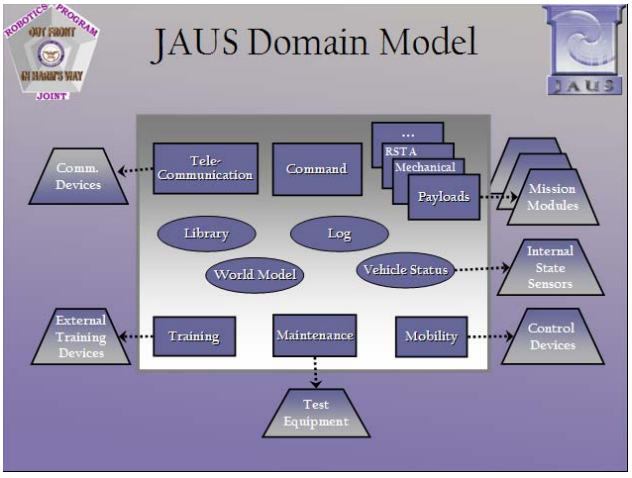

Figure 8: JAUS Domain Model for Open Architecture Autonomous Vehicles (OUSD).

4.2 Move to 3-D Space

In order to effectively establish the capabilities of any of these GAs, 3-D space is highly desirable. With this comes validation that the system will perform given the increased kinematics, and dynamics with 6 degrees of freedom. Several different models are available that model UAVs, aircraft and helicopters.

In Slear (2006) and Corner (2004), a 3-D model is used to simulate UAV movement. In this case it employed a static formulation of the swarm based upon Reynolds rules to move on a path defined a priori with a mission planning evolutionary algorithm. SO rules defined here could be applied to that swarm. This simulation uses SkyView to visualize the movement of the swarm in 3D down the apriori path. In addition, this simulation has been integrated into the SPEEDES Parallel and Discrete Event Simulator (PDES) (Melendez 2006, Slear 2006).

4.3 Development of a Robust Set of BAs

One of the first steps to advance this tool would be an augmentation of BAs creating a robust set for all situations. Although focused BAs would be directed by the accepted Techniques Tactics and Procedures (TTP) of the Air Force rated world (pilots), more can be developed to allow the swarm more automation for more generic situations. The possibilities include: specified search algorithms, rendezvous, path following, and engagement optimization.

4.4 Inclusion of an Effective SO Planning Algorithm Applied through Distributed Communications

In order to fashion a coherent database throughout the swarm, a Z-Tree algorithm needs to be implemented to distribute and retrieve the data easily (Seetharaman 1997). If communication routes are created in a similar way, they can be used to communicate information. This would allow for effective planning or target recognition via distributed data storage across the swarm network. The Z-Tree heuristic would also maintain the one of the main intents of swarm theory, and create no single point of failure.

4.5 Building an Effective User Interface for the Development of Rules and Optimization

Currently, Swarmfare still requires a programmer to modify and control the simulation's flexibility. At some point an extended User Interface needs to be developed for those manipulations. A central control point is needed to access the many visualizations (Path, Real-time, Field-plot, BehaviorSet, Search, and Sensor viewers), develop new scenarios, modification of flight kinematics variables and more. Note that the Behavior Archetypes (BA) are developed by the GA, which means little user control past the number of BAs allowed.

4.6 Communication Robustness/Accuracy

More robust explicit communications should be implemented very carefully so that it does not start to encroach on the SO capabilities. It can be exploited in the transfer of information for more abstract computation and direction. For instance, it could be used for mapping the search space, target recognition on a distributed database across the swarm, or man in the loop controls. Again, the explicit and implicit communication roles must clearly be defined in order not to disrupt the functionality of the SO of the swarm.

4.7 Man in Loop Control

Another feature that will be important for the simulation to implement is the inclusion of user directed instructions. This could come in the form of apriori path guidance or real-time mission revectoring. The latter of which will take a tremendous amount of communication backbone and support. The other concern to address when imple-

menting this is the level of control is how the instructions are implemented. If the instructions are too low level, then the SO behavior could be effected, and if too highly abstracted, the system many not be able to effectively incorporate those directions. Many have started to work on this including, Cummings, who directly address the problem of levels of SO (more generally Automation) (Cummings 2004). Figure 9 shows the levels of Automation as defined by Sheridan and Verplank (1987).

Automation Level	Automation Description
1	The computer offers no assistance: human must take all decision and actions.
2	The computer offers a complete set of decision/action alternatives, or
3	narrows the selection down to a few, or
4	suggests one alternative, and
5	executes that suggestion if the human approves, or
6	allows the human a restricted time to veto before automatic execution, or
7	executes automatically, then necessarily informs humans, and
8	informs the human only if asked, or
9	informs the human only if it, the computer, decides to.
10	The computer decides everything and acts autonomously, ignoring the human.

Figure 9: Sheridan and Verplank (1978) levels of automation.

When exploring the requirements of this capability the intent must be considered. Many possibilities can spring from the intent. Is the controller able to virtually move from UAV to UAV to directly control them? Would it be beneficial to have a human controlled leader? Can the user establish control in an outside UAV that would pull the swarm into two parts to accomplish a side task? These types of questions all lead to more and deeper questions, finally resulting in very different implementations.

5 SUMMARY

The SWARMFARE system is a work in progress. Currently there research is ongoing for the development of further SO capabilities. These requirements include optimized reconnaissance, distributed data storage, classification schemes, distributed image recognition and hierarchal approaches to integrating the swarm into existing communications infrastructure. The development of the GUI will centralize all of the functionality of the system into one user friendly interface. A path planning Multi-Objective Evolutionary Algorithms (MOEA) has been developed for UAV swarms using a PDES and the inclusion of these in a SO environment is ongoing (Slear). Efforts into JAUS compliance and robust communications are on the near horizon as well.

In summary, the SWARMFARE Simulation is a capable tool in which research of the governing rule sets of swarm behavior can evolve. Its parallel and distributed architectures, 2D visualization and analytical data output make this a strong tool. The fundamental GA implemented through SO framework is its biggest attraction. Through thorough design of the basic rule set of SO, the system can be shaped as needed to most UAV SO swarming experiments. The system can continue to evolve incorporating more user friendly interfaces, added inter-agent communications support, and man-in-the-loop controls. As this tool continues to evolve, the SO behaviors and emergent structures formulated and tested can be used to define the capabilities of the next generation autonomous vehicles and military UAVs.

ACKNOWLEDGMENTS

The sponsor of this research is the Air Force Research Laboratory (AFRL) Information Directorate and the Sensors Directorate. Mike Foster is the lead of the Virtual Computing Laboratory of AFRL/SN.

REFERENCES

Bonabeau, E., M. Dorigo, and G. Theraulaz. 1999. *Swarm intelligence: from natural to artificial systems.* New York: Oxford University Press.

Camazine, S., J.-L. Deneoubourg, N. R. Franks, J. Sneyd, G. Theraulaz and E. Bonabeau. 2003. *Self-Organization in Biological Systems*. USA: Princeton University Press.

Cantu-Paz, E. 2005. Theory of parallel genetic algorithms. in *Parallel Metaheuristics: A New Class of Algorithms*, ed. E. Alba. New Jersey: Wiley-Interscience.

Cottam, R., W. Ranson, and R. Vounckx. 2003. Autocreative hierarchy II: dynamics self-organization, emergence and level-changing. In *International Conference on Integration of Knowledge Intensive Multi-Agent Systems,* 766-773.

Corner, J. 2004. *Swarming Reconnaissance using Unmanned Aerial Vehicles in a Parallel Discrete Event Simulation*. M.S. Thesis. Air Force Institute of Technology, Dayton, OH.

Cummings, M. L. 2004. Human supervisory control of swarming networks. 2nd Annual Swarming Autono-

mous Intelligent Networked Systems Conference. Massachusetts Institute of Technology.

Gat, E. 1997. On three-layer architectures. In *Artificial Intelligence and Mobile Robots*, ed. R. P. Bonnasso, D. Kortenkamp, and R. Murphy, MIT/AAAIPress.

Heylighen, F. 2006. Self-organization and complexity in the natural sciences. Principia Cybernetica Web. Available via <http://pespmc1.vub.ac.be/compnats.html> [accessed August 17, 2007].

Lua, C. A., K. Altenburg and K. E. Nygard. 2003. Synchronized multi-point attack by autonomous reactive vehicles with local communication. In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, 95-102.

Melendez, K., J. Slear, and G. Lamont. 2006. Parallel UAV swarm simulation with optimal route planning. *Summer Simulation Conference*.

Nowak, D. 2006. Self organized systems in the real world and their computational implementation. Technical Report. Air Force Institute of Technology, Dayton, OH.

Office of the Under Secretary of Defense (OUSD) for Acquisition, Technology and Logistics. Joint architecture for unmanned systems. Available via <http://www.jauswg.org/> [accessed January 17, 2007].

Price, I. 2006. *Evolving self-organized behavior for homogeneous and heterogeneous UAV or UCAV swarms.* M.S. Thesis. Air Force Institute of Technology, Dayton, OH.

Price, I. and G. Lamont. 2006. GA directed self-organized search and attack UAV swarms. In *Proceedings of the 2006 Winter Simulation Conference*, ed. L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, 1307-1315. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Reynolds, C. W. 1987. Flocks, herds, and schools: a distributed behavioral model. In *Computer Graphics 4*, ed. M. C. Stone, 25-34. SIGGRAPH.

Reynolds, C. 1999. Steering behaviors for autonomous characters. *Game Developers Conference*.

Rosenblatt, J. K. 1997. DAMN: a distributed architecture for mobile navigation. In *Proceedings of the AAAI Spring Symposium on Lessons Learned from Implemented Software Architectures for Physical Agents*. Stanford, CA.

Rubio, J. S., J. Vagners, R. Rydsyk. 2004. Adaptive path planning for autonomous UAV oceanic search missions. *AiAA 1st Intelligent Systems Technical Conferenc.*

Seetharaman, G. and B. Zavidovique. 1997. Image processing in a tree of peano coded images. In *Proceedings of the 1997 Computer Architectures For Machine Perception (CAMP '97)*. IEEE Computer Society, Washington, DC, 229.

Sheridan, T. B. and W. Verplank. 1978. *Human and computer control of undersea teleoperators.* Cambridge, MA: Man-Machine Systems Laboratory, Department of Mechanical Engineering, MIT.

Slear, J. 2006. *AFIT UAV swarm mission planning and simulation system.* M.S. Thesis, AFIT/GCE/ENG/06-08. Graduate School of Engineering and Management, Air Force Institute of Technology, Dayton, OH.

UAVForum. 2005. Predator. Available via <http://www.uavforum.com/vehicles/production/predator.htm> [accessed August 17, 2007].

AUTHOR BIOGRAPHIES

DUSTIN J. NOWAK is a Masters student in the Department of Electrical and Computer Engineering, Graduate School of Engineering at Air Force Institute of Technology, with a research focus on artificial intelligence specifically SO in UAVs. His e-mail address is <Dustin.Nowak@afit.edu>.

IAN C. PRICE is the chief of high performance computation at AEDC, Bldg 100, suite A325704 CS/CL, Arnold AFB, TN 37389. He graduated from Air Force Institute of Technology with a Master's degree in 2006. His e-mail address is <ian.price@arnold.af.mil>.

GARY B. LAMONT is a Professor in the Department of Electrical and Computer Engineering, Graduate School of Engineering at the Air Force Institute of Technology specializing in evolutionary algorithms and high performance computation. He is an IEEE Life Senior Member. His e-mail address is <gary.lamont@afit.edu>.