A CASE FOR ARTIFICIAL INTELLIGENCE IN AUTONOMOUS AIR VEHICLES

Joseph D. Mainardi

GENERAL DYNAMICS
Convair Division
5001 Kearny Villa Road
Mail Zone 44-6980
San Diego, CA 92138
(619) 694-9120

INTRODUCTION

Smart weapons, like the Tomahawk missile, are already a reality. Because smart mission management and search analysis are a few of the new challenges facing developers, the use of Artificial Intelligence (AI) technologies will be a viable alternative to conventional software systems that will be developed for use in an autonomous air vehicle (AAV).

The mission management task involves the command and control for high-level functions of on-board software subsystems. An AI solution would be appropriate for use in this type of decision support system. The leading candidates for this task are based on: decision trees, a hybrid frame/object-based & rule-based system, or a hybrid data-driven & goal-directed system. Another task is priority area search terrain analysis, where a terrain grid is subdivided into cells and analyzed to determine the most likely locations for targets. This will require an analysis of a combination of positive and negative conditions, which is well suited to AI technology. The most likely candidate for this task is an advanced feature recognition algorithm. To support future implementation in an embedded, on-board environment, the software will be developed using the Ada language.

This paper will present an overview of these two tasks, in the form of an alternative approach to conventional software solutions. In addition, the issues of basic design using AI methodologies, tool selection and verification & validation will be discussed.

BACKGROUND

As part of the on-going effort to provide smart weaponeering for AAV's, General Dynamics Convair Division has been contracted to meet specific objectives to reach that goal. One of the objectives is to demonstrate the ability to perform real-time search and recognition under realistic target deployment scenarios and flight conditions. A specific objective that is well suited for AI technologies is in providing smart search for an AAV, which will include utilization of "a priori" data on target locations and target deployment tactics.

This paper was funded under Independent Research & Development (IR&D) for the Department of Defense. The Aerospace Industries Association of America (AIAA)

considers the implementation of real-time, embedded AI technology to be an important technological addition to the design and development of future weapons systems. In a document released by the AIAA in 1989, AI technology was identified as one of the eight key technologies for the 1990's. Their selection of AI was founded on the basis of AIAA internal studies, and was chosen on the basis of: broadest application base, highest leverage and greatest potential payoff. While there are successful non real-time AI based systems in place, the goal here is to demonstrate real-time AI based systems, as an alternative to conventional software systems, that will exhibit the benefits of AI technology within the scope of AAV's.

There are two specific applications that show promise as AI based applications. One of the tasks was to provide on-board mission management support. This system will manage the overall decision making required for an on-board computer system, while controlling interface and component subsystems that require real-time performance in a semi-autonomous environment. The other task was to provide a way to perform priority area search terrain analysis in real-time. This system will provide a mechanism that will permit a terrain grid to be subdivided into cells and analyzed, in order to determine the most likely locations for targets or threats.

Mission Manager Unit - MMU

The overall decision management system is called the Mission Manager Unit. Because many of the decisions to be made are required in real-time, the MMU will have to provide guaranteed response times. Due to the mix of on-board conventional and AI based software, integrating numeric and symbolic computing will be necessary. A typical decision management system provides command and control functions for all of the on-board software subsystems. Most of the command and control will occur within the MMU, while the remaining command and control will occur within interface subsystems associated with the component subsystems. The MMU will have command and control of a mix of component (conventional) subsystems and interface (AI technology) subsystems. Figure 1 depicts a possible generic final implementation for the MMU.

A component subsystem, or reactive agent, is any conventional software that is designed or implemented outside of the specific requirements of the MMU, and whose communication information is required for operations related to the MMU. These subsystems solve hard real-time problems, which refer to those deadlines which, if not met, will likely lead to catastrophic failures (loss of human life, permanent hardware failure, vehicle destruction, etc.). An interface subsystem, or cognitive agent, is any AI based software that is designed or implemented as a specific requirement of the MMU, and is used to interface between the MMU and a component subsystem. These subsystems solve soft real-time problems, which refer to those deadlines which, if not met, will rarely lead to catastrophic failures. Interface subsystems are most often used under the following conditions: (1) communication between the MMU and a component subsystem is not feasible, (2) the use of a knowledge based system makes good sense as an interface between the MMU and a component subsystem, or (3) the MMU rules directly related to a component subsystem grow to the point that they would create difficulties for the MMU in achieving real-time performance levels.

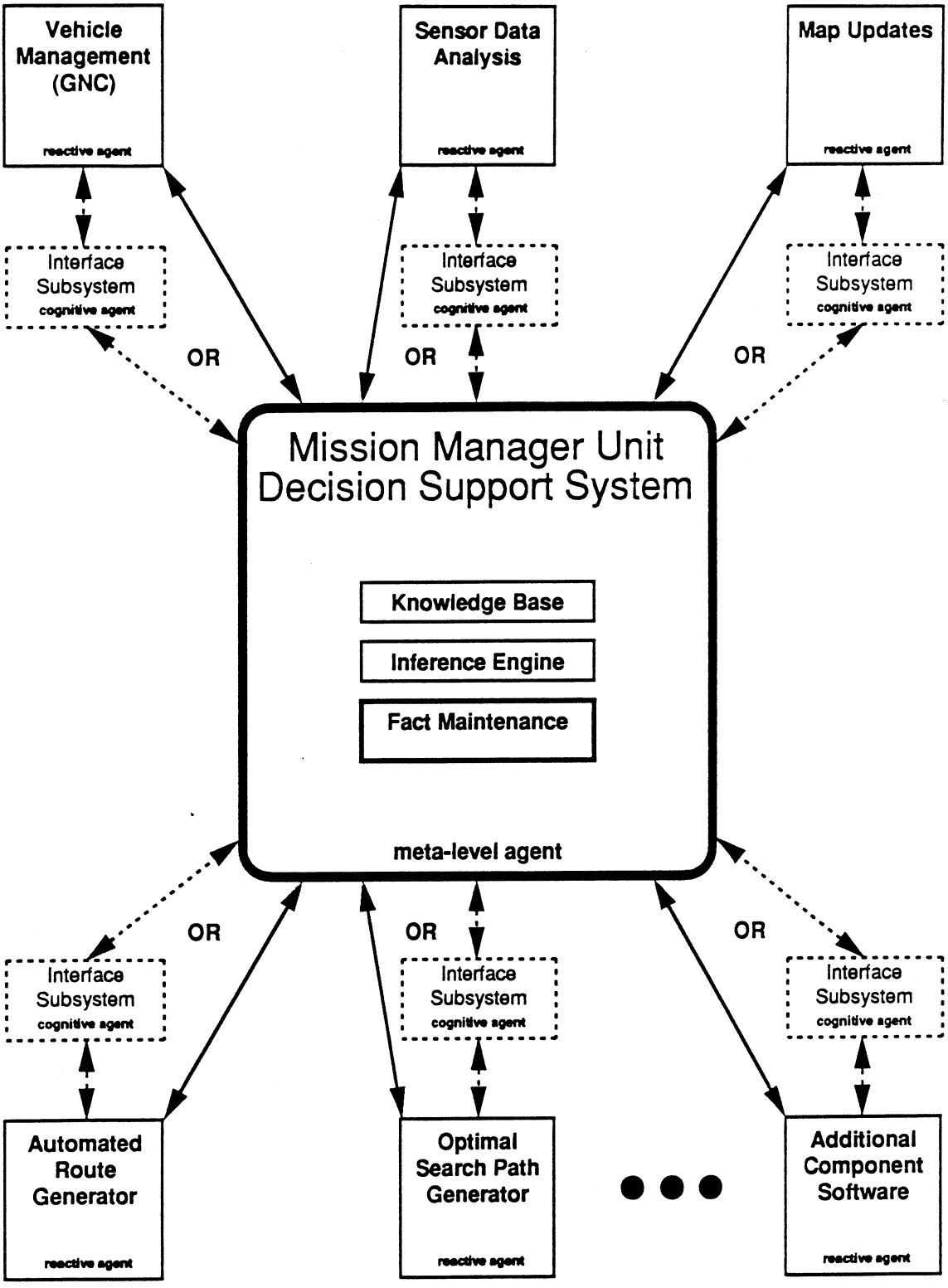

Figure 1, MMU final implementation

The MMU and the component and interface subsystems are considered agents and they are initially created as stubs. An agent is any fundamental active entity associated with the MMU, for both component and interface subsystems. A stub is either data or code the emulates or simulates the functions and responses of an actual agent. Stubs will be replaced whenever a component subsystem or interface subsystem has been completely tested, verified and validated for use in the overall system. The MMU is considered a meta-level agent, which performs command and control operations such as: task assignment to lower-level agents, reconciliation of conflicting recommendations, data input control, scheduling and time management.

Agents may be perceived as knowledge sources, which can be any type of software program of widely varying size and complexity, as they either produce or modify knowledge. Most agents, particularly interface subsystems, will have capabilities, interests and acquaintances. A capability is an output information field from an agent, and is used by other agents that have both an interest in the information and is an acquaintance of the agent providing the output. An interest is an input information field that is used by an agent to perform specific tasks, and can only use the information if is an acquaintance of the agent providing the information. An acquaintance is a representation of a relationship between two or more agents that are associated with common information fields.

The approach taken to create the components of smart search in a real-time environment was to create a knowledge based system that would provide a distributed base for the on-board decision making process. There are many advanced technology concepts that are applicable to the MMU task. Intelligent information transfers to and from the MMU are useful and necessary. This information transfer is in the form of information fields, and can be passed between the MMU and either a component subsystem or an interface subsystem. Each MMU subsystem agent will be separately evaluated, and will be designed and developed based on that evaluation. Some subsystem agents will require AI based solutions, while others will not require any form of AI technology. The concept of semi-autonomy permits the component subsystems to control their specific environments, while still being responsive to command and control information from the MMU. This information is transmitted between the MMU and component subsystems or interface subsystems.

Due to the complex interrelationships between the agents, the distributed AI concept of multi-agent systems will be used. The multi-agent system will comprise intelligent behavior among the semi-autonomous subsystems. These agents will coordinate knowledge, data and goals in order to take action or solve problems. Interaction and communication will be routed through the MMU, so there will be no interconnections between the other agents. It is possible to create interconnections between other agents, but this would probably push us beyond a real-time window for all but the most simple cases. We will use a component based framework, as we are designing and building a conglomerate of mutually dependent tasks which will be decomposed into appropriate subtasks. Because there will be occasions where we will have the need to go beyond the surface understanding of a problem and understand the underlying domain knowledge, deep knowledge will be used. Because it is less biased toward direct use, deep knowledge allows us to create reusable software for other subsystems or projects.

The concept of approximate processing will be used for the MMU. Approximate processing can be used on both data and knowledge, and will make real-time solutions possible. The need for approximate processing is because current methods in smart weapons mission planning have not yet reached an appropriate level of expertise. An example of the need for approximate processing would be in the area of real-time mission replanning. By being able to approximate the decision making process, some of the detailed analysis can be avoided. Under certain circumstances, approximate processing may be better suited to our purposes than exact processing, as solutions might not be possible if exact processing was used. These approximate processing activities will be consistent with exact processing activities, and both strategies will be combined at a later date. Exact processing will replace approximate processing when the level of mission planning expertise is raised to a level that permits the replacement to take place. Knowledge approximation can be used until such time that exact knowledge strategies are available. It would not be unusual to leave an approximate processing system in place even after exact knowledge was available, as the approximate knowledge may be sufficient, and may perform better than an exact knowledge counterpart.

In order to guarantee real-time command and control, fact maintenance is required for the the MMU and each interface subsystem. This is a requirement that will help to avoid data degradation. Fact maintenance should be invoked whenever: (1) a milestone has occurred (area scanned, target destroyed, etc.), or (2) certain data or facts are no longer considered useful or credible.

We will be developing software based on at least one of three knowledge representation concepts: decision trees, a hybrid goal-directed & data-driven system, or a hybrid frame/object-based & rule-based system. In so much as the information about these knowledge representation concepts is widely available, I will not go into detail about the concepts. Instead, I will mention the possible candidate software tools for each concept. In the area of decision trees, The "Knowledge Shaper" tool (from Perceptics) will generate optimized procedural Ada code, based on one or more irreducible decision trees. This tool creates the decision trees with or without the implementation of user designed cost or control biases. We would need to create the software that would invoke these generated processes. The book "Artificial Intelligence with Ada" (by Louis Baker, 1989, McGraw Hill Publishers) has examples and source code for both goal-directed and data-driven systems. Our internal design and development effort would require an integration template joining the two types of systems. For the hybrid frame/object-based & rule-based system, the "ART/Ada" tool (from Inference) integrates frames & objects with rules. Also, the "Artificial Intelligence with Ada" book has examples and source code for this type of system. This would be an internal design and development effort, and would only be used if there was an agent that required a hierarchical inheritance structure.

Priority Area Search Terrain Analysis - PASTA

Another task was to provide a way to perform real-time priority area search terrain analysis, in order to determine the most likely locations for targets or threats. PASTA will permit an autonomous air vehicle to rank and rate the cells of a terrain grid, based on a list of specified vehicles. The PASTA agent will be an interface subsystem of the MMU, and will have its own specific subgoals. It will identify possible target areas, based on "a priori" information about the terrain grid in question.

A straightforward algorithm is used to help determine the most likely locations for targets or threats. Vehicle specific continuous functions are required to quantify relationships based on terrain slopes and terrain cell proximities. An example of a terrain slope function is now described. *For an SS-21 TEL (Transporter Erector Launcher), the terrain slope value is between 0.0 and 1.0 inclusive, where the continuous function produces values such that 0.0 equals 90 degrees incline, 0.5 equals a 45 degree incline and 1.0 equals a 0 degree incline.* The equation looks like: $\mathbf{TS_{SS\text{-}21\ TEL}} = (\mathbf{90\text{-}incline_{[deg]}}) / \mathbf{90}$. Two possible examples of terrain cell proximities are now described. The first example is for a UAZ-469 (Jeep). *The proximity value for a tree cell near a road cell is 0.0 to 1.0 inclusive, where the continuous function produces values such that 0.0 equals a distance of 250 meters or more from the tree cell to the nearest road cell, 0.5 equals a distance of 125 meters from the tree cell to the nearest road cell,and 1.0 equals a distance of 0 meters from the tree cell to the nearest road cell.* The equation looks like: $\mathbf{TP_{SS\text{-}21\ TEL[tree\text{->}road]}} = (\mathbf{250\text{-}distance_{[meters]}}) / \mathbf{250}$. The second example is also for a UAZ-469. *The proximity value for a tree cell near an open field cell is 0.0 to 1.0 inclusive, where the continuous function produces values such that 0.0 equals a distance of 100 meters or more from the tree cell to the nearest open field cell, 0.5 equals a distance of 50 meters from the tree cell to the nearest open field cell,and 1.0 equals a distance of 0 meters from the tree cell to the nearest open field cell.* The equation looks like: $\mathbf{TP_{SS\text{-}21\ TEL[tree\text{->}field]}} = (\mathbf{100\text{-}distance_{[meters]}}) / \mathbf{100}$.

For each of the steps presented in the algorithm example on the following pages, the *italics text* offers a verbal description of the equations. The first step is to read in all of the negative effector condition thresholds and rules for each type of vehicle that is to be identified (**STEP 1**). This step uses "a priori" knowledge about all vehicle types to determine if it is necessary to look in a cell for a particular type of vehicle. By using rules and thresholds, a negative value (0.0) or positive value (1.0) can be assigned to any vehicle likelihood. The next step is to read in potential positive effector conditions and likelihoods (**STEP 2**). This step uses "a priori" knowledge about all vehicle types to determine the potential positive effectors for all of the vehicle types.

After reading in the negative and potential positive effectors, the next step is to read in the cell information and likelihoods of the terrain cells (**STEP 3** shows a subset of the cell input data). This cell information is also "a priori" knowledge about the terrain; updated terrain information can also be read in at this step. The next step is to compute the negative effector condition likelihood for each class & cell combination (**STEP 4**). This step is actually the execution of the rules from STEP 1, above. The fifth step in this algorithm is to compute the potential positive effector condition likelihood for each vehicle class and cell combination (**STEP 5**). This step multiplies

the values of each component in STEP 2 to the related component values for each cell in STEP 3, and then dividing that number by the number of common components. For comparison purposes, the MAX option is also displayed. The last step is to compute the likelihoods for each vehicle type in all cells (**STEP 6**). This step simply multiplies that vehicle related values from STEP 4 and STEP 5 to produce the probabilities. Once again, the MAX option is also displayed.

Figure 2 shows a diagrammed view of a possible analysis. The shaded "probability" squares reflect pre-determined probability levels for target or threat likelihood. For example, an **Excellent** value would be considered a 75% to 100% target or threat probability, and an **Average** value would be considered a 50% to 62.5% target or threat probability. The numbers inside the cells represent a ranking of the search area for a particular vehicle type (a UAZ-469 Jeep, for example).

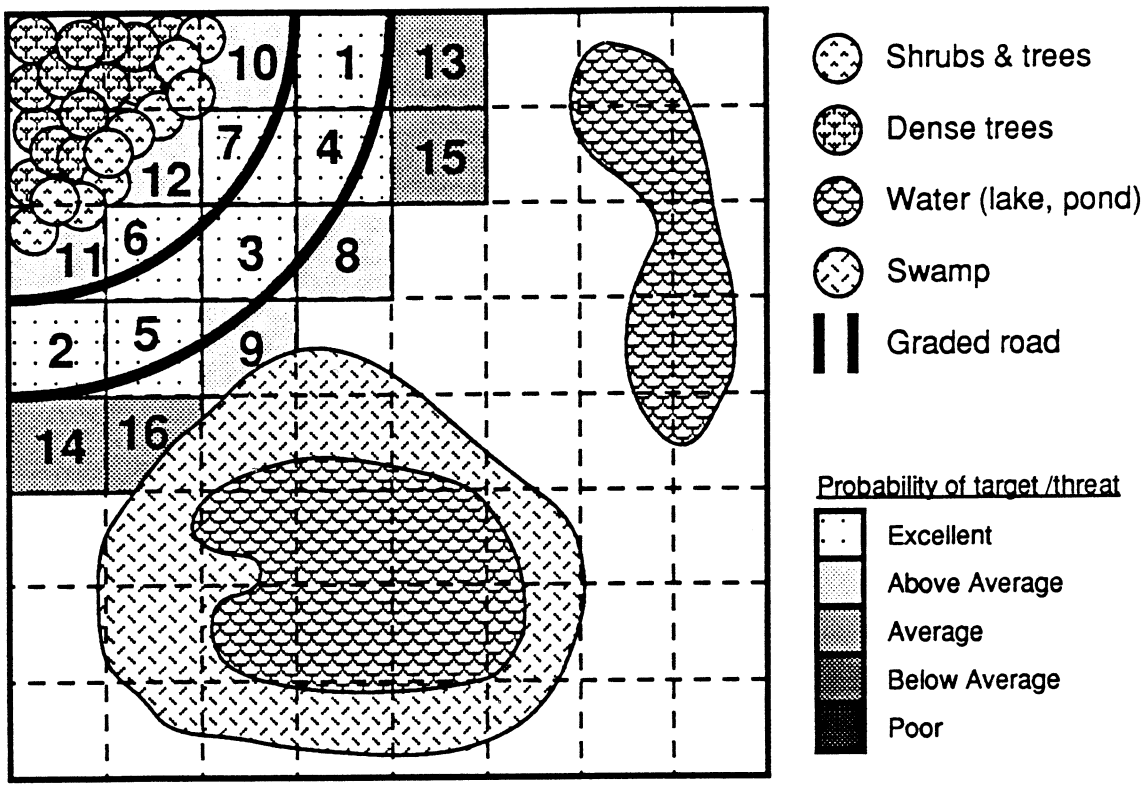

Figure 2, Terrain cell rankings (with shaded probability levels)

An example of the PASTA algorithm is now presented. Note the following references: [1] **SS-21 TEL** is code for a Transporter Erector (Missile) Launcher, [2] **BTR-60PA** is code for an Armored Command Vehicle, and [3] **UAZ-469** is code for a Jeep. Also note that the **->** symbol represents relativity between cells. For example, **tree->road** refers to the relative distance from a tree dominated cell to the closest road dominated cell.

STEP 1 - Read in the negative effector condition thresholds & rules.

SS21-TEL thresholds

$NTL_{SS\text{-}21\ TEL}$[field] = [1.0]
$NTL_{SS\text{-}21\ TEL}$[tree] = [1.0] *threshold likelihood for SS-21 TEL in a tree cell is 1.0*
$NTL_{SS\text{-}21\ TEL}$[urban] = [1.0]
$NTL_{SS\text{-}21\ TEL}$[railroad] = [1.0]
$NTL_{SS\text{-}21\ TEL}$[water] = [1.0]
$NTL_{SS\text{-}21\ TEL}$[tree->field] = [0.8] *threshold likelihood for SS-21 TEL in a tree cell and "near" a field is 0.8*
$NTL_{SS\text{-}21\ TEL}$[tree->road] = [0.6]
$NTL_{SS\text{-}21\ TEL}$[field->tree] = [0.8]
$NTL_{SS\text{-}21\ TEL}$[terrain slope] = [0.9]

SS21-TEL rules

$NL_{SS\text{-}21\ TEL}$ = 1.0
If urban >= $NTL_{SS\text{-}21\ TEL}$[urban] then $NL_{SS\text{-}21\ TEL}$ = 0.0
If railroad >= $NTL_{SS\text{-}21\ TEL}$[railroad] then $NL_{SS\text{-}21\ TEL}$ = 0.0
If water >= $NTL_{SS\text{-}21\ TEL}$[water] then $NL_{SS\text{-}21\ TEL}$ = 0.0
If field >= $NTL_{SS\text{-}21\ TEL}$[field] *and* field->tree < $NTL_{SS\text{-}21\ TEL}$[field->tree] then $NL_{SS\text{-}21\ TEL}$ = 0.0
If tree >= $NTL_{SS\text{-}21\ TEL}$[tree] *and* tree->field < $NTL_{SS\text{-}21\ TEL}$[tree->field] then $NL_{SS\text{-}21\ TEL}$ = 0.0
If tree >= $NTL_{SS\text{-}21\ TEL}$[tree] *and* tree->road < $NTL_{SS\text{-}21\ TEL}$[tree->road] then $NL_{SS\text{-}21\ TEL}$ = 0.0
If terrain slope < $NTL_{SS\text{-}21\ TEL}$[terrain slope] then $NL_{SS\text{-}21\ TEL}$ = 0.0

UAZ-469 thresholds

$NTL_{UAZ\text{-}469}$[field] = [1.0]
$NTL_{UAZ\text{-}469}$[tree] = [1.0]
$NTL_{UAZ\text{-}469}$[urban] = [1.0]
$NTL_{UAZ\text{-}469}$[railroad] = [1.0]
$NTL_{UAZ\text{-}469}$[water] = [1.0]
$NTL_{UAZ\text{-}469}$[swamp] = [1.0]
$NTL_{UAZ\text{-}469}$[tree->field] = [0.5]
$NTL_{UAZ\text{-}469}$[tree->road] = [0.7]
$NTL_{UAZ\text{-}469}$[field->tree] = [0.5]

UAZ-469 rules

$NL_{UAZ\text{-}469}$ = 1.0
If urban >= $NTL_{UAZ\text{-}469}$[urban] then $NL_{UAZ\text{-}469}$ = 0.0
If railroad >= $NTL_{UAZ\text{-}469}$[railroad] then $NL_{UAZ\text{-}469}$ = 0.0
If water >= $NTL_{UAZ\text{-}469}$[water] then $NL_{UAZ\text{-}469}$ = 0.0
If swamp >= $NTL_{UAZ\text{-}469}$[swamp] then $NL_{UAZ\text{-}469}$ = 0.0
If field >= $NTL_{UAZ\text{-}469}$[field] *and* field->tree < $NTL_{UAZ\text{-}469}$[field->tree] then $NL_{UAZ\text{-}469}$ = 0.0
If tree >= $NTL_{UAZ\text{-}469}$[tree] *and* tree->field < $NTL_{UAZ\text{-}469}$[tree->field] then $NL_{UAZ\text{-}469}$ = 0.0
If tree >= $NTL_{UAZ\text{-}469}$[tree] *and* tree->road < $NTL_{UAZ\text{-}469}$[tree->road] then $NL_{UAZ\text{-}469}$ = 0.0

STEP 2 - Read in the potential positive effector conditions & likelihoods.

$PL_{SS\text{-}21\ TEL}$[tree->field] = [0.9] *likelihood of SS-21 TEL in a tree cell and "near" a field is 0.9*
$PL_{SS\text{-}21\ TEL}$[tree->road] = [0.7]
$PL_{SS\text{-}21\ TEL}$[field->tree] = [0.9]
$PL_{SS\text{-}21\ TEL}$[field->road] = [0.6]
$PL_{SS\text{-}21\ TEL}$[road->tree] = [0.6]
$PL_{SS\text{-}21\ TEL}$[road->field] = [0.8]
$PL_{SS\text{-}21\ TEL}$[road] = [0.7]
$PL_{SS\text{-}21\ TEL}$[field] = [0.5]

STEP 2 - cont.

$PL_{UAZ\text{-}469}$[tree->field] = [0.7]
$PL_{UAZ\text{-}469}$[tree->road] = [0.8]
$PL_{UAZ\text{-}469}$[field->tree] = [0.9]
$PL_{UAZ\text{-}469}$[field->road] = [0.9]
$PL_{UAZ\text{-}469}$[road->tree] = [0.6]
$PL_{UAZ\text{-}469}$[road->field] = [0.9]
$PL_{UAZ\text{-}469}$[road] = [0.9]
$PL_{UAZ\text{-}469}$[field] = [0.5]

STEP 3 - Read in the cell information & likelihoods of three random cells.

$cell_2$[road] = [1.0] *road likelihood is 1.0 for this cell*
$cell_2$[tree] = [0.0]
$cell_2$[field] = [0.0]
$cell_2$[water] = [0.0]
$cell_2$[swamp] = [0.0]
$cell_2$[urban] = [0.0]
$cell_2$[railroad] = [0.0]
$cell_2$[terrain slope] = [0.9] *terrain slope likelihood is 0.9 for this cell*
$cell_2$ [tree->field] = [0.0]
$cell_2$ [tree->road] = [0.0]
$cell_2$ [field->tree] = [0.0]
$cell_2$ [field->road] = [0.0]
$cell_2$ [road->tree] = [0.9]
$cell_2$ [road->field] = [0.9]

$cell_{52}$[road] = [0.0]
$cell_{52}$[tree] = [0.0]
$cell_{52}$[field] = [0.0]
$cell_{52}$[water] = [1.0]
$cell_{52}$[swamp] = [0.0]
$cell_{52}$[urban] = [0.0]
$cell_{52}$[railroad] = [0.0]
$cell_{52}$[terrain slope] = [0.1]
$cell_{52}$ [tree->field] = [0.0]
$cell_{52}$ [tree->road] = [0.0]
$cell_{52}$ [field->tree] = [0.0]
$cell_{52}$ [field->road] = [0.0]
$cell_{52}$ [road->tree] = [0.0]
$cell_{52}$ [road->field] = [0.0]

$cell_{64}$[road] = [0.0]
$cell_{64}$[tree] = [0.0]
$cell_{64}$[field] = [1.0]
$cell_{64}$[water] = [0.0]
$cell_{64}$[swamp] = [0.0]

STEP 3 - cont.

$cell_{64}$ [urban] = [0.0]
$cell_{64}$ [railroad] = [0.0]
$cell_{64}$ [terrain slope] = [0.6] *terrain slope likelihood is 0.6 for this cell*
$cell_{64}$ [tree->field] = [0.0]
$cell_{64}$ [tree->road] = [0.0]
$cell_{64}$ [field->tree] = [0.7]
$cell_{64}$ [field->road] = [0.5]
$cell_{64}$ [road->tree] = [0.0]
$cell_{64}$ [road->field] = [0.0]

STEP 4 - Compute the negative effector condition likelihood for each class & cell combination.

$cell_2NL_{SS\text{-}21\ TEL} = 1.0$
$cell_2NL_{UAZ\text{-}469} = 1.0$

$cell_{52}NL_{SS\text{-}21\ TEL} = 0.0$
$cell_{52}NL_{UAZ\text{-}469} = 0.0$

$cell_{64}NL_{SS\text{-}21\ TEL} = 0.0$
$cell_{64}NL_{UAZ\text{-}469} = 1.0$

STEP 5 - Compute the potential positive effector condition likelihood for each class & cell combination.

$cell_2PL_{SS\text{-}21\ TEL} = [(0.6 * 0.9) + (0.8 * 0.9) + (0.7 * 1.0)] / 3 = 0.65$ {MAX = 0.72}
$cell_2PL_{UAZ\text{-}469} = [(0.6 * 0.9) + (0.9 * 0.9) + (0.9 * 1.0)] / 3 = 0.75$ {MAX = 0.9}

$cell_{52}PL_{SS\text{-}21\ TEL} = 0.0$
$cell_{52}PL_{UAZ\text{-}469} = 0.0$

$cell_{64}PL_{SS\text{-}21\ TEL} = [(0.6 * 0.5) + (0.9 * 0.7) + (0.5 * 1.0)] / 3 = 0.48$ {MAX = 0.63}
$cell_{64}PL_{UAZ\text{-}469} = [(0.9 * 0.5) + (0.9 * 0.7) + (0.5 * 1.0)] / 3 = 0.53$ {MAX = 0.63}

STEP 6 - Compute likelihoods from steps 4 & 5.

$cell_2CL_{SS\text{-}21\ TEL} = 1.0 * 0.65 = 0.65$ {MAX = 0.72} *likelihood of SS-21 TEL in cell 2 is 0.65 {MAX likelihood is 0.72}*
$cell_2CL_{UAZ\text{-}469} = 1.0 * 0.75 = 0.75$ {MAX = 0.9}

$cell_{52}CL_{SS\text{-}21\ TEL} = 0.0 * 0.0 = 0.0$
$cell_{52}CL_{UAZ\text{-}469} = 0.0 * 0.0 = 0.0$

$cell_{64}CL_{SS\text{-}21\ TEL} = 0.0 * 0.48 = 0.0$
$cell_{64}CL_{UAZ\text{-}469} = 1.0 * 0.53 = 0.53$ {MAX = 0.63}

CONCLUSIONS

We have demonstrated that there will be tasks where the use of AI technology will be a viable alternative to conventional software systems for autonomous air vehicles. Two such tasks have been discussed in this paper. The mission management task, where control of the high-level functions of the on-board software subsystems will occur, and the priority area search terrain analysis task, where a terrain grid is subdivided into cells and analyzed to determine the most likely locations for targets or threats. While there is significant interest in incorporating real-time AI technology to make smart weapons a reality before the turn of the century, the ability to provide expertise for these tasks is not as readily available. Until such time, the best that can be done is to

emulate the expertise. This can be accomplished by using selected information from current autonomous air vehicle systems.

One of the issues that confronts AI based systems is in verification & validation (V&V). While conventional, procedural Ada code is fairly easy to verify & validate, most AI based software has not fared as well in the past. This is not to say that AI based software can not be verified or validated. The major obstacle confronting the developers of tasks such as the two in this paper, is to generate an end product that conforms to V&V standards. The software development of the tasks presented in this paper lend themselves to a converted Ada based procedural code end-product. This fact alone greatly increases our ability to verify & validate the software. By developing modular code whose end-product is procedural in nature, the process of proving the correctness and functionality will be easier than with "pure AI" systems. It is this combination of advanced AI techniques and V&V acceptability that will permit AI based systems to move into the mainstream of real-time embedded computer systems.