Proceedings of the IEEE-OES Autonomous Underwater Vehicles 2012, September 2012, Southampton, England

An Implemented Universal Mission Controller with Run Time Ethics Checking for Autonomous Unmanned Vehicles—a UUV Example

Don Brutzman, Robert McGhee, Duane Davis
Modeling, Virtual Environments and Simulation (MOVES) Institute, Naval Postgraduate School, Monterey, California USA
brutzman@nps.edu, robertbmcghee@gmail.com, dtdavi1@nps.edu

***Abstract*—The authors have been involved for several decades in the development and testing of both remotely controlled and autonomous subsea and ground vehicles. This experience has led us to view autonomous mobile robot control problems from both a bottom up and a top down perspective. Specifically, in our work, we have developed and tested a three-level software architecture called Rational Behavior Model (RBM), in which a top (strategic) level mission control finite state machine (FSM) orders the rational execution, at an intermediate (tactical) level, of vehicle behaviors in such a way as to carry out a specified mission. This implementation experience and these principles have led us to believe that human-like intelligence and judgment are not required to achieve a useful operational capability in autonomous mobile robots. Furthermore, we are convinced that a primitive but useful type of robot ethical behavior can also be attained, even in hazardous or military environments, without invoking concepts of artificial intelligence. To support our views, we present a software invention called a mission execution engine (MEE), implemented in the Prolog logic programming language. This MEE can be shown to represent an extension of the idea of a universal Turing machine and is therefore well grounded in existing mathematical automata theory. We further show how human readable mission orders, also written in Prolog, can specialize an MEE to any desired mission control FSM. An important aspect of our work is that mission orders can be tested exhaustively in human executable form before being translated into robot executable form. This provides the kind of transparency and accountability needed for after action review of missions, and possible legal proceedings in case of loss of life or property resulting from errors in mission orders.**

***Index Terms*—AUV, UUV, autonomous robot control, ethical robot control, Turing machine.**

I. Introduction

There is at present underway a revolutionary change in the role of unmanned vehicles in warfare and other high risk environments [1, 2]. In the case of unmanned undersea vehicles (UUVs), some are true autonomous robots, capable of operating for months at sea, without human intervention [3]. In a recent publication [4], the authors propose an extension of the most general theoretical model for computation, a *Turing machine* (TM) [5], into a broader class of automata, that we call *Mission Execution Automata* (MEA). Among other possible applications, MEA are intended provide a mathematically grounded basis for control of long duration autonomous robot missions, with an ability to deal with contingencies arising during extended mission execution. Specifically, an MEA consists of a *mission specific* finite state machine (FSM) [5], provided with one or more *external agents*, each with an ability to sense the environment in some well-defined and limited way, to respond to commands issued by the FSM, and to answer a finite set of predetermined queries from the FSM. When the external agent is an incremental tape recorder, a Turing machine results. In [6], an implemented *universal* Turing machine is presented as an MEA, thereby proving that Turing machines constitute a proper subclass of MEA.

Analogous to a universal Turing machine [5], we have found it useful to define and realize a *universal Mission Execution Engine* (MEE) that can imitate any mission specific MEA by executing a given set of *mission orders* [4]. These orders are analogous to the formal written mission orders typically provided to the commander of a manned submarine. They are also analogous to, but more general than, the *machine description* part of the *tape* of a universal Turing machine [5]. For UUVs operating in high risk environments, we believe that it is a moral imperative that such orders be subjected to exhaustive human testing in simulation form before recoding into robot executable form. When such testing has been completed, then the senior mission specialist participating in the testing can formally approve these orders as executable specifications for the subsequent generation of robot mission orders by robot specialists. He then could be legally solely *accountable* for any errors in mission orders. In addition, when mission orders are written in appropriately structured Prolog [7], we claim that they can be read *declaratively* by non-programmers, thus providing the kind of *transparency* needed for after action review, and possible legal proceedings relating to loss of life or property.

After formal approval of mission orders in human testable form, the next step can involve a sequence of specialists taking the code in a controlled and testable way toward full robot executable form, thus creating a *chain of command*, with a single designated individual accountable for code correctness at each level of abstraction, where such code constitutes executable and testable specifications for refinement in the level below it [8]. This process is facilitated by adopting the Rational Behavior Model (RBM) software architecture [9], in

Goal 1. Proceed to Area A and search the area. If the search is successful execute Goal 2. If the search is unsuccessful, execute Goal 3.

Goal 2. Obtain an environment sample from Area A. If the sample is obtained, execute Goal 3. If the sample cannot be obtained, proceed to recovery position to complete the mission.

Goal 3. Proceed to Area B and search the area. Upon search success or failure, execute Goal 4.

Goal 4. Proceed to Area C and rendezvous with UUV-2. Upon rendezvous success or failure, proceed to recovery position to complete the mission.

Fig. 1. Example manned submarine area search and sample mission orders expressed in structured natural language.

which *strategic level* code for mission control is realized as an FSM, specifically as an MEA.

Below the strategic level in RBM lies the *tactical* level in which commands from the MEA controlling the mission are refined to *execution* level commands resulting in actual vehicle maneuvers and environmental sensing [9]. The tactical level also replies to queries from the strategic level by integrating sensed information from the execution level. The remainder of this paper develops, in computer simulation form, an example mission for a UUV using Allegro Prolog [10] for MEA realization. The main contribution of this work is to show that finite state machine theory and predicate logic [7, 11] are sufficient for providing useful degree of *machine intelligence* to an autonomous mobile robot. We claim that *artificial intelligence* [12, 13], mimicking human behavior in the control of manned submarine missions, is not required to achieve "field programmability" in a UUV, including a primitive form of run time ethics checking. In this paper we develop and test a prototypical "area search and sample mission" to illustrate our approach and support this contention.

Experimental results, both for UUVs and ground robots operating with RBM software in natural outdoor environments, are already available and confirm the effectiveness of this software architecture [9, 14, 15]. There are no questions of *feasibility* associated with the adoption of RBM and MEA for autonomous mobile robot control software.

II. Example Structured Natural Language Mission Orders and Associated State Graph

Figure 1 presents a prototypical set of mission orders for a (simplified) "area search and sample" mission to be conducted by a manned submarine [4].

As a step toward formalizing the above orders for MEA execution, it is useful to construct a corresponding state graph as shown in Fig. 2. In examining this figure, it should be noted that two of the phases enclosed by ovals are *terminal* phases (or states) that have no successor states. In addition, there is a "Start" phase that has no predecessor state. These phases are implicit in the word statement of Fig. 1, but, as will be seen, need to be made explicit for MEA execution [4].

Before proceeding further, it is important to determine that the senior mission specialist agrees that Fig. 2 correctly captures the intent of the word statement of Fig. 1. For this paper, the authors have exhaustively reviewed these two figures, and are in agreement that the graph of Fig. 2 is correct. This having been accomplished, Fig. 2 can now be "animated" to actually issue commands and to query a human "tactical officer" to carry out a simulated mission. In this paper, this animation is accomplished by a MEE "invented" by us and encoded in Prolog [4]. This MEE, together with its mission orders is presented in the next section of this paper.

III. Universal Mission Execution Engine and Prolog Mission Orders

Figure 3 below, taken from [4], defines a universal MEE in executable form using Allegro Prolog. To understand the code, it is necessary to recognize that an arrow symbol designates rule definition and that the expression inside the first parenthesis is the *rule head*, while the remainder of the expression is the *rule body* [7]. Thus, the first line of code can be read declaratively as: "A mission is executed if it is initialized and successive phases are then executed until done." The next line says: "Initializing a mission is accomplished if the current phase is set to 1." The third line says "Consult the fact database to find the value of the 'current phase' variable, and then execute that phase."

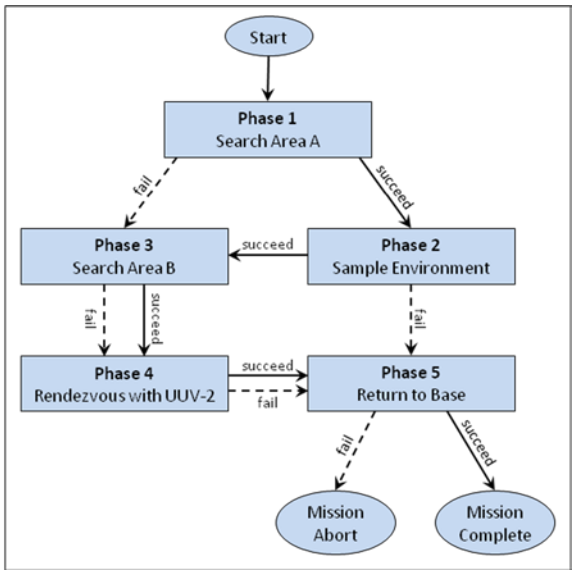

Fig. 2. State graph for prototypical unmanned undersea vehicle (UUV) area search and sample mission.

```
;C:/Documents and Settings/mcghee/My Documents/Tech Reports/Ethical MEA/mission-controller.cl"

;This code was written in Allegro ANSI Common Lisp, Version 8.2.

;Allegro Prolog uses Lisp syntax. Rule head is first expression following "<--" symbol. Rule
;body is rest of expressions. Subsequent definitions of rule use "<-" symbol to avoid overwrite.

;Mission orders must be compiled and saved in location specified by your (edited) version of “load”
;command below. If "mission-controller.cl" has not been previously compiled, it may be necessary to
;open it in a new Allegro Editor window to avoid "name conflict error" response from compiler.

(require :prolog) (shadowing-import '(prolog:==)) (use-package :prolog) ;Start Prolog.

(load "C:/Documents and Settings/mcghee/My Documents/Tech Reports/Ethical MEA/mission-orders.fasl")

;Facts

(<-- (current_phase 0)) ;Start phase.

;Mission execution rule set

(<-- (execute_mission) (initialize_mission) (repeat) (execute_current_phase) (done) !)
(<-- (initialize_mission) (abolish current_phase 1) (asserta ((current_phase 1))))
(<-- (execute_current_phase) (current_phase ?x) (execute_phase ?x) !)
(<-- (done) (current_phase 'mission_complete))
(<- (done) (current_phase 'mission_abort))

;Human external agent communication functions

(<-- (negative nil))
(<- (negative n))
(<-- (affirmative ?x) (not (negative ?x)))
(<-- (report ?C) (princ ?C) (princ ".") (nl))
(<-- (command ?C) (princ ?C) (princ "!") (nl))
(<-- (ask ?Q ?A) (princ ?Q) (princ "?") (read ?A))

;Test function (illustrates format for calling for mission execution from Lisp)

(defun tm () (?- (execute mission)))
```

Fig. 3. Prolog code for Universal Mission Execution Engine (MEE) with human external agent communication functions.

Finally, the last two lines state that mission execution is over if the current phase is “mission complete” or “mission abort”. Remarkably, these few lines define, in executable form, means for sequencing phases for a very wide range of mission types, including all possible Turing machine missions [6]. The reader is referred to [4, 7, 13] for a more complete discussion of Prolog syntax and semantics.

The following Fig. 4, adapted from [4], presents Prolog mission orders for the “area search and sample” mission of Fig. 2. Comparing Fig. 2 to Fig. 4, it can be seen that execution of Phase 1 involves issuing the command to a human tactical officer to “Search Area A”, and a branch to Phase 2 if the search succeeds and Phase 3 if it does not. A similar interpretation can be made for all other phases in this mission. Again, this comparison was made for all phases of the selected mission, and the authors agree on the correctness of the above mission orders. In examining this code, it should be noted that the commands issued by States 3 and 4 have been prefixed to include the word “attempt”. This was done in order to emphasize that subsequent commands are not conditioned on the outcome of such an attempt.

IV. VALIDATION OF MISSION ORDERS THROUGH EXHAUSTIVE TESTING

Fortunately, since an MEA is by definition a finite state machine, providing that it is loop free, it can be tested exhaustively by answering the queries arising during human mission execution in every possible way. Figure 5 presents partial results of such a test for the mission orders of Fig. 4. To conserve space in this paper, user responses to prompts 3 through 5 have been omitted. Also, for ease of understanding these results, user responses have been manually highlighted in bold.

The authors recognize that obtaining and interpreting the above kind of results can be challenging, especially when more phases are involved than in this example. However, we have collectively reviewed Fig. 5, along with the results of replying to queries with the other three possible answer sequences (ynn, yny, and yyn), and agree that results obtained correspond to what we intended when writing the above mission orders.

An important question is: “How do we know that the above test is exhaustive?” After all, the mission orders include three questions, each of which can be answered in two ways, so

```
(require :prolog) (shadowing-import '(prolog:==)) (use-package :prolog) ;Start Prolog.

;Utility functions

(<-- (change_phase ?old ?new) (retract ((current_phase ?old))) (asserta ((current_phase ?new))))


;Mission specification

(<-- (execute_phase 1) (command "Search Area A") (ask "Search successful" ?A) (affirmative ?A)
     (change_phase 1 2))
(<- (execute_phase 1) (change_phase 1 3))

(<- (execute_phase 2) (command "Sample environment") (ask "Sample obtained" ?A) (affirmative ?A)
    (change_phase 2 3))
(<- (execute_phase 2) (change_phase 2 5))

(<- (execute_phase 3) (command "Attempt Area B search") (change_phase 3 4))

(<- (execute_phase 4) (command "Attempt rendezvous with UUV2") (change_phase 4 5))

(<- (execute_phase 5) (command "Return to base") (ask "At base" ?A) (affirmative ?A)
    (change_phase 5 'mission_complete) (report "Mission succeeded"))
(<- (execute_phase 5) (change_phase 5 'mission_abort) (report "Mission failed"))
```

Fig. 4. Prolog mission orders for human execution of prototypical UUV area search and sample mission.

```
CG-USER(1): (?- (execute_mission))
Search Area A!
Search successful?n
Attempt Area B search!
Attempt rendezvous with UUV2!
Return to base!
At base?n
Mission failed.
Yes

No.
CG-USER(2): (?- (execute_mission))
Search Area A!
Search successful?n
Attempt Area B search!
Attempt rendezvous with UUV2!
Return to base!
At base?y
Mission succeeded.
Yes

No.
CG-USER(6): (?- (execute_mission))
Search Area A!
Search successful?y
Sample environment!
Sample obtained?y
Attempt Area B search!
Attempt rendezvous with UUV2!
Return to base!
At base?y
Mission succeeded.
Yes

No.
```

Fig. 5. Partial results from exhaustive human interactive testing of area search and sample mission orders (user responses in bold).

shouldn't there be eight cases to be tested? The answer to this question is "no," the reason being that some missions are executed with less than three responses being invoked. Specifically, the "Sample environment" command associated with Phase 2 of the mission is not issued in either of these cases because, as can be seen from the mission orders, Phase 2 is entered only upon success of Phase 1. This results in a total of only six cases rather than eight.

Some deeper questions arose from our discussions of the results of Fig. 5. For example, referring to Fig. 2, why should the vehicle return to base upon failing to get a sample from Area A? Wouldn't it be better for it to go on and attempt to sample Area B? For that matter, if the purpose of the mission is to obtain a sample, why not return to base after *success* of sampling in Area A rather than in the event of *failure* of this phase? Was this a mistake in the original human natural language mission definition presented in Fig. 1? Only appeal to a higher authority can answer this kind of question. However, this is an additional value of human mission rehearsal that we feel will prove to be essential to some types of UUV missions. In particular, we believe that no potentially lethal robot mission orders should be executed by an operational vehicle until they have passed the kind of exhaustive "Turing test" [13] illustrated by Fig. 5 above. When all questions have been resolved concerning exhaustive testing results, then in the most meaningful sense of the term, the Prolog mission code can said to have been "proved correct".

As a final remark, the above exhaustive testing of mission execution is possible only because the control flow graph of Fig. 3 is loop free. If mission orders contain a loop, then the possibility of an infinite number of test cases arises. As demonstrated in [6, 14], this can be dealt with by utilizing a *loop count* or *time out* failure mode in tactical level behavior implementation.

V. Autonomous Execution

As explained above, execution of all code presented so far in this paper requires the participation of a human "tactical officer". This is not unrealistic for some applications. Specifically, if such a person were in control of a remotely operated vehicle (ROV or "drone"), then he or she would be able to execute commands from an MEA, and also to respond to queries during the execution of a real mission. However, that is not our goal here. Rather, in this paper, we are primarily interested in totally autonomous execution. To accomplish this in simulation form requires the development of both tactical level and execution level software. That is, there must be code written to interpret commands and queries from the MEA to the simulated robot as calls to implemented functions at these levels. Since the Allegro dialect of Prolog used in this paper for MEA realization is based on Common Lisp, this is the natural language to use for this purpose. Such a simulation for the mission under consideration here is presented in [8]. Typical results are presented in Fig. 6.

It is important to realize that the choice of Allegro Prolog and Common Lisp for the work presented so far in this paper is based on a desire for clarity of exposition and to allow convenient code development and execution on a personal computer. Readers interested in such execution are referred to [8] for code and detailed instructions.

Actual robot experiments using the RBM architecture [9, 14, 15] have used both Prolog and other languages. Nevertheless, regardless of final implementation languages, at the present time we believe that Prolog provides the best widely available basis for development and verification of mission orders by mission specialists, who are not necessarily programmers. Better than any other method we know of, this approach provides transparency, as well as accountability, in assigning legal responsibility for issuance of executable mission orders as specifications for real time code development. We also believe that the incremental code development methods used in [8] are equally well applicable to other languages. The use of detailed graphical representations of both vehicles and their environments adds much to the viability of such an approach to code development and validation for control of autonomous mobile robots [16].

```
CG-USER(1): (tm)
Search Area A!
Robot report: Search Area A succeeded.
Sample environment!
Robot report: Sample environment succeeded.
Search Area B!
Robot report: Search Area B succeeded.
Rendezvous UUV2!
Robot report: Rendezvous UUV2 succeeded.
Return to base!
Robot report: Return to base succeeded.
Mission succeeded.
Yes

No.
CG-USER(2): (tm)
Search Area A!
Robot report: Search Area A failed.
Search Area B!
Robot report: Search Area B succeeded.
Rendezvous UUV2!
Robot report: Rendezvous UUV2 failed.
Return to base!
Robot report: Return to base failed.
Mission failed.
Yes

No.
```

Fig. 6. Examples of mission logs resulting from execution of fully autonomous mission simulation code.

VI. Run Time Ethics Checking

As mentioned in the introduction to this paper, a primitive form of run time ethics checking can be included in the software architecture we are advocating. Specifically, a mission specialist could add "robot ethical constraints" to structured natural language robot orders such as those of Fig. 1. A simple example is provided by Fig. 7 below.

Of course these rules are very simple compared to typical human *rules of engagement*. However, this paper is not aimed at human mission execution. Rather, our goal is to show how useful missions can be specified and conducted by autonomous mobile robots without requiring that they possess humanlike artificial intelligence. Whether or not this is so will eventually require physical experiments involving realistic ethical constraints on robot behavior. At this point in time we merely wish to show how code for such experiments can be written in computer simulation form. Incorporating the Fig. 4 constraints into the code of Fig. 7 yields the revised mission orders of Fig. 8. Partial results from the beginning and end of testing of all possible user responses to the queries issued by this code are depicted in Fig. 9.

The results make use of the beginning of a "just say no" exhaustive testing algorithm. Specifically, each query is answered "no" unless to do so would repeat a previous pattern. If this is the case, then the query is answered "yes". Surprisingly, following this algorithm, it turns out that only 19 distinct answer sequences are possible in executing the mission orders of Fig. 8. This is in contrast to the 256 possible sequences of arbitrary "yes" or "no" answers to eight questions. The reason for this is of course, that most answer sequences terminate with success or failure after many fewer than 8 queries. The authors have examined all 19 answer sequences and are in agreement that the given robot ethical constraints have been correctly incorporated into the previously verified mission orders of Fig. 4.

Of course, as with the code of Fig. 4, this code is suitable

Constraint 1: If ethical search of Area A is not possible, go to Goal 4.

Constraint 2: If ethical execution of Goal 3, 4, or 5 is not possible, abort the mission.

Fig. 7. Example robot ethical constraints for area search and sample mission.

```
(require :prolog) (shadowing-import '(prolog:==)) (use-package :prolog) ;Start Prolog.


;Utility functions

(<-- (change_phase ?old ?new) (retract ((current_phase ?old))) (asserta ((current_phase ?new))))
(<-- (ethical_abort ?x) (ask "Ethical execution possible" ?A) (not (affirmative ?A))
     (change_phase ?x 'mission_abort) (report "Mission aborted. Unethical command"))
(<-- (ethical_branch ?old ?new) (ask "Ethical execution possible" ?A) (not (affirmative ?A))
     (change_phase ?old ?new) (report "Phase aborted. Unethical command"))


;Mission specification

(<-- (execute_phase 1) (command "Search Area A") (phase_completed 1))
(<-- (phase_completed 1) (ethical_branch 1 4))
(<- (phase_completed 1) (command "Execute command") (ask "Successful-1" ?A)
    (affirmative ?A) (change_phase 1 2))
(<- (phase_completed 1) (change_phase 1 3))

(<- (execute_phase 2) (command "Sample environment") (phase_completed 2))
(<- (phase_completed 2) (ethical_abort 2))
(<- (phase_completed 2) (command "Execute command") (ask "Successful-2" ?A)
    (affirmative ?A) (change_phase 2 3))
(<- (phase_completed 2) (change_phase 2 5))

(<- (execute_phase 3) (command "Search Area B") (phase_completed 3))
(<- (phase_completed 3) (ethical_abort 3))
(<- (phase_completed 3) (command "Execute command") (change_phase 3 4))

(<- (execute_phase 4) (command "Rendezvous UUV2") (phase_completed 4))
(<- (phase_completed 4)(ethical_abort 4))
(<- (phase_completed 4) (command "Execute command") (change_phase 4 5))

(<- (execute_phase 5) (command "Return to base") (phase_completed 5))
(<- (phase_completed 5) (ethical_abort 5))
(<- (phase_completed 5) (ask "At base" ?A) (affirmative ?A)
    (change_phase 5 'mission_complete) (report "Mission succeeded"))
(<- (phase_completed 5) (change_phase 5 'mission_abort) (report "Mission failed"))
(<- (phase_completed 2) (change_phase 2 5))

(<- (execute_phase 3) (command "Search Area B") (phase_completed 3))
(<- (phase_completed 3) (ethical_abort 3))
(<- (phase_completed 3) (command "Execute command") (change_phase 3 4))

(<- (execute_phase 4) (command "Rendezvous UUV2") (phase_completed 4))
(<- (phase_completed 4)(ethical_abort 4))
(<- (phase_completed 4) (command "Execute command") (change_phase 4 5))

(<- (execute_phase 5) (command "Return to base") (phase_completed 5))
(<- (phase_completed 5) (ethical_abort 5))
(<- (phase_completed 5) (ask "At base" ?A) (affirmative ?A)
    (change_phase 5 'mission_complete) (report "Mission succeeded"))
(<- (phase_completed 5) (change_phase 5 'mission_abort) (report "Mission failed"))
```

Fig. 8. Mission orders for area search and sample mission with run time ethics checking included.

only for execution by a human tactical officer. Presumably such a person would have sufficient knowledge of the tactical situation (when controlling a remotely operated vehicle or during code debugging by simulation means), and of human rules of engagement, to make correct decisions in deciding if ethical execution of a given command is possible or not. At this point in time, encoding of such judgments for robot execution cannot be expected to give results as good as human judgment. Much more research will be needed before this becomes possible. However, the authors wish to point out that all such future code development would be at the tactical level, and would not change the mission orders of Fig. 8. Moreover, further consideration may show that even very primitive robotic ethical constraints, such as those included in this paper, may prove to be better than none. Only real operational experience can show whether this is true or not.

VII. Summary and Conclusions

The purpose of this paper is to show a means by which the well understood tasking of missions for manned submarines by

```
CG-USER(1): (tm)
Search Area A!
Ethical execution possible?n
Phase aborted. Unethical command.
Search Area B!
Ethical execution possible?n
Mission aborted. Unethical command.
Yes

No.

CG-USER(19): (tm)
Search Area A!
Ethical execution possible?y
Execute command!
Successful-1?y
Sample environment!
Ethical execution possible?y
Execute command!
Successful-2?y
Search Area B!
Ethical execution possible?y
Execute command!
Rendezvous UUV2!
Ethical execution possible?y
Execute command!
Return to base!
Ethical execution possible?y
At base?y
Mission succeeded.
Yes

No.
```

Fig. 9. Partial results of exhaustive testing of area search and sample mission with run time ethics checking.

means of written mission orders can be applied to autonomous unmanned undersea vehicles. Our methodology makes use of a software invention called a *Universal Mission Execution Engine* (MEE). In the implementation presented here, we use the Prolog logic programming language for clarity and to allow mission orders written in a particular specified format to be read by human mission experts who may not be computer programmers. Further, these Prolog orders can be executed in simulation form (and ultimately in actual robots) without recoding once verified for "correctness" by a human tactical officer. Our approach allows for a limited form of run time ethics testing by a real vehicle during actual execution of a potentially lethal mission. An important feature of our work is that artificial intelligence concepts are not required in the development and execution of such orders. Rather, it is shown by computer simulation studies that finite state machine theory and predicate calculus are sufficient for this purpose.

REFERENCES

[1] Singer, P.W., *Wired for War*, Penguin Books, London, England, 2009.

[2] Arkin, R.C., *Governing Lethal Behavior in Autonomous Robots*, CRC Press, New York, 2009.

[3] Clooney,L., and Webb,D., "Increasing the Operational Envelope for the Slocum Electric Glider", *Proc. Of 17th International Symposium on Unmanned Untethered Submersible Technology*, Portsmouth, NH, August, 2011.

[4] McGhee, R. B., Brutzman, D. P., and Davis, D. T., "A Universal Multiphase Mission Execution Automaton (MEA) with Prolog Implementation for Unmanned Untethered Vehicles", *Proc. Of 17th International Symposium on Unmanned Untethered Submersible Technology*, Portsmouth, NH, August, 2011. Available at https://savage.nps.edu/AuvWorkbench/website/documentation/papers/papers.html

[5] Minsky, M.L., *Computation: Finite and Infinite Machines*, Prentice Hall, 1967.

[6] McGhee, R.B., Brutzman, D.P., and Davis, D.T., *A Taxonomy of Turing Machines and Mission Execution Automata with Lisp/Prolog Implementation*, Technical Report NPS-MV-11-002, Naval Postgraduate School, Monterey, CA 93943, June, 2011. Available at https://savage.nps.edu/AuvWorkbench/website/documentation/reports/reports.html

[7] Rowe, N.C., *Artificial Intelligence through Prolog*, Prentice Hall, Englewood Cliffs, NJ 07632, 1988.

[8] McGhee, R.B., Brutzman, D.P., and Davis, D.T., *Recursive Goal Refinement and Iterative Task Abstraction for Top-Level Control of Autonomous Mobile Robots by Mission Execution Automata – A UUV Example*, Technical Report NPS-MV-12-002, Naval Postgraduate School, Monterey, CA 93943, August, 2012. Available at https://savage.nps.edu/AuvWorkbench/website/documentation/reports/reports.html

[9] Brutzman, D., et al, "The Phoenix Autonomous Underwater Vehicle", *Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems*, Ch. 13, pp. 323-360, ed. by Kortenkamp, D., et al, MIT Press, Cambridge, MA 02142, 1998.

[10] Franz, Inc., Allegro Prolog Online Documentation, 2011. Available at www.franz.com/support/documentation/current/doc/prolog.html

[11] Kohavi, Z., and Niraj, K.J., *Switching and Finite Automata Theory*, McGraw Hill, 2010.

[12] Nilsson, N.J., *The Quest for Artificial Intelligence*, Cambridge University Press, New York, 2010.

[13] Russell, S.J., and Norvig, P., *Artificial Intelligence: A Modern Approach*, Prentice Hall, 1995.

[14] Marco, D.B., Healey, A.J., and McGhee, R.B., "Autonomous Underwater Vehicles: Hybrid Control of Mission and Motion", *Autonomous Robots 3*, pp. 169-186, 1996.

[15] Davis, D.T., Brutzman, D.P., and Becker, W.J., "Facilitation of Autonomous Vehicle Coordination through an XML-Based Vehicle-Independent Control Architecture", *Proc. of the 16th International Symposium on Unmanned Untethered Submersible Technology*, Durham, NH, August, 2009. Available at https://savage.nps.edu/AuvWorkbench/website/documentation/papers/papers.html

[16] Davis, D.T., and Brutzman, D.P., "The Autonomous Unmanned Vehicle Workbench: Mission Planning, Mission Rehearsal, and Mission Replay Tool for Physics-Based X3D Visualization", *Proc. Of 14th International Symposium on Unmanned Untethered Submersible Technology*, Durham, NH, August, 2005. Available at https://savage.nps.edu/AuvWorkbench/website/documentation/papers/papers.html