Ethical Mission Definition and Execution for Maritime Robots Under Human Supervision

Don Brutzman, *Member, IEEE*, Curtis L. Blais, *Member, IEEE*, Duane T. Davis, and Robert B. McGhee, *Fellow, IEEE*

***Abstract*—Experts and practitioners have worked long and hard toward achieving functionally capable robots. While numerous areas of progress have been achieved, ethical control of unmanned systems meeting legal requirements has been elusive and problematic. Common conclusions that treat ethical robots as an always-amoral philosophical conundrum requiring undemonstrated morality-based artificial intelligence are simply not sensible or repeatable. Patterning after successful practice by human teams shows that precise mission definition and task execution using well-defined, syntactically valid vocabularies is a necessary first step. Addition of operational constraints enables humans to place limits on robot activities, even when operating at a distance under gapped communications. Semantic validation can then be provided by a Mission Execution Ontology to confirm that no logical or legal contradictions are present in mission orders. Thorough simulation, testing, and certification of qualified robot responses are necessary to build human authority and trust when directing ethical robot operations at a distance. Together these capabilities can provide safeguards for autonomous robots possessing the potential for lethal force. This approach appears to have broad usefulness for both civil and military application of unmanned systems at sea.**

***Index Terms*—Autonomous vehicles, mission execution automata (MEA), mission execution ontology (MEO), robot ethics.**

I. Nature of Ethical Missions

EXPERTS and practitioners have worked long and hard toward achieving functionally capable robots. While numerous areas of progress have been achieved, progress in ethical control of unmanned systems has been elusive and problematic. Common conclusions that treat ethical robots as an always-amoral philosophical conundrum or requiring undemonstrated morality-based artificial intelligence (AI) are simply not sensible or repeatable. For better or worse, actors around the world are rapidly designing and deploying mobile unmanned systems to augment human capabilities. Thus, theory must rise to meet practice.

This work adapts policies and procedures for ethical responsibility and authority that have been proven to work in collaborative military operations, even across varying cultures and platforms. Patterning after successful practice by human teams shows that precise mission definition and task execution can provide safeguards for autonomous robots or human–robot teams possessing potentially lethal capabilities. Since lethality is not limited to military weapons but can also include navigational interference and vehicle collisions, and since many robots are capable of carrying out well-defined tasks regardless of their internal software architecture, this approach appears to have broad usefulness for civil application of unmanned systems as well.

Experience and experimentation across four decades of robotic and military operations inform this work. The authors first look at unmanned capabilities and limitations, along with real-world exemplars of how humans delegate command responsibility and authority. Robot mission tasks and goals can be clearly specified and refined with corresponding degrees of internal control supervision occurring, in the case of the exemplar discussed here as part of a three-layer software architecture. The Autonomous Vehicle Command Language (AVCL) [1]–[4] allows expressing such mission constructs in a formal yet human-understandable way, matching the repertoires of most human-driven and robot-supervised vehicles. Adding well-defined prerequisite constraints (permission, restriction, and required human intervention) can supplement mission orders in context of each individual task, providing an ethical basis for unmanned system tasking that matches human understanding of similar responsibilities. Careful structuring of a mission execution automation (MEA) demonstrates a theoretically sound and scalable basis to this approach. The functional vocabulary is intentionally restricted to the well-understood mission capabilities of humans and robots so that broad compatibility by many robots is possible. Strict-subset vocabularies might alternatively implement these atomic concepts using slightly different syntax, but the core concepts must remain consistent.

Modeling, simulation, and visualization have enabled extensive testing of mission operations, building human confidence in well-defined task orders. The Extensible Markup Language (XML) validation of AVCL tasks confirms syntactical correctness of mission orders, but more is needed. The authors therefore have created a Mission Execution Ontology (MEO) based on principles of description logics (DLs), and implemented using Semantic Web languages. This ontology is used to confirm that mission definitions are also semantically complete, including ethical constraints whenever appropriate. Such premission verification of mission completeness is analogous to

Manuscript received January 29, 2017; accepted December 6, 2017. Date of publication January 23, 2018; date of current version April 12, 2018. This work was supported in part by the Naval Postgraduate School Consortium for Robotics and Unmanned System Education and Research and in part by the Office of the Secretary of Defense Joint Ground Robotics Enterprise. *(Corresponding author: Donald Brutzman.)*

Guest Editor: W. Kirkwood.

The authors are with the Naval Postgraduate School, Monterey, CA 93943 USA (e-mail: brutzman@nps.edu; clblais@nps.edu; dtdavis@nps.edu; robertbmcghee@gmail.com).

Digital Object Identifier 10.1109/JOE.2017.2782959

chain-of-command human review of operations orders that already occurs before coordinated team operations.

A long trail has led to this point, inspired by many sources but driven by a need to implement practical constraints on unmanned systems lethality. A feasible path forward now exists [5]. Semantic coherence of mission orders for humans and robots working together can be achieved if tasks include ethical constraints that define acceptable operational prerequisites for remote action. Current project conclusions show that much work remains for ethical control of robots, but progress is indeed possible and quite encouraging. The authors believe that ethical human supervision of semiautonomous unmanned systems is feasible today and widely repeatable in a practical manner.

II. Considering Critical Challenges

The idea of intelligent robots emerged from and developed in the minds of artists and dreamers long before the prevailing technology was capable of supporting its underlying premises. First imagined using the term "robot" in the Czech play *Rossum's Universal Robots* [6], these intelligent humanoid machines were relegated primarily to the realm of science fiction in the first half of the twentieth century. Even so, the ethical ramifications of mobile (and potentially lethal) machines capable of human-like intelligence and actions were readily apparent, and seemingly reasonable ethical frameworks, most notably Azimov's Three Laws of Robotics [7], were devised to govern intelligent robot operation. As science fiction aficionados are well aware, then and now, these frameworks were rife with loopholes and unanticipated subtleties that inevitably led to their downfall.

The advent of digital computing, the emergence of AI as an academic discipline, and the simultaneous incorporation of both into a variety of robotic devices have brought these ethical concerns to the forefront of academic and practical debate. Moreover, the ready availability of this technology to governments, corporations, research entities, and individuals has made this issue one of broad societal importance. From robotic vacuum cleaners to armed military drones, intelligent robotic technology has insinuated itself into aspects of our lives that were not previously imagined. One implication of this ubiquity is that questions of legal and moral responsibility will not be answered by a set of fixed "laws" and cannot be regulated into irrelevance through government action, just as is the case of endeavors involving humans.

Nevertheless, a number of important observations can be made, which are as follows.

1) *Predictability:* Robots essentially perform exactly as programmed to perform in a given situation. Predictability is independent of the intent of the programmer, the understanding of the operator, and any anthropomorphic bias of observers. Thus, a trustworthy robot must be sufficiently competent to perform assigned tasks.
2) *Authority:* Apparent intelligence notwithstanding, a robot is an inanimate object. Thus, moral responsibility for the consequences of a robot's actions cannot be assigned to the robot. Decision-making authority must be performed by qualified, well-informed humans.
3) *Responsibility:* Direct responsibility for the outcomes of robot activity must accompany authority, and must be assignable to a specific human entity. For robot ethics to bear any tangible meaning, ultimate moral and legal accountability must reside with the human programmers, manufacturers, operators, and leadership. Deliberate care must be taken when giving orders to robots, just as is already given for orders to humans.
4) *Liability:* The assignment of liability (whether legal or moral) in any circumstance is premised on the assumption that the involved parties are in a position to reasonably foresee the outcomes for which they are being held responsible. Liability accompanies authority and responsibility.

These observations are fairly widely accepted, but nevertheless can lead ethicists to different conclusions. In debating military use of autonomous systems, for instance, Rob Sparrow of the International Committee for Robot Arms Control uses *Jus en Bello* requirements to argue that the military use of lethal robots is inherently unethical because robots cannot be held accountable for their actions [8]. Ronald Arkin, on the other hand, accepts the premises of Sparrow's argument but comes to the opposite conclusion—that if an autonomous system is capable of making a lethal decision more reliably than a human, then it is inherently unethical to *not* use that system [9].

Notwithstanding disagreements over military use of autonomous robots, these observations can form a common basis that provides a framework for ethical operation of intelligent robots afloat. This approach is feasible with current technologies and without a requirement for black-box AI "ethical controllers" that do not integrate well with specialized software schemes and inevitably lead to second-guessing, obfuscation, and uncertainty. Further, this paradigm is potentially applicable not only to military operations (lethal or otherwise) but also to other employment of robotic systems, where questions of ethical operation and responsibility arise [10] [11].

Human-directed unmanned systems must follow the well-established legal principle of vicarious liability [12] in both military and civilian applications, where operators can be held morally and legally responsible for all outcomes from a robot's activities if they are in a position to foresee those outcomes. That is, operators can be held responsible for undesirable outcomes that they are in a position to prevent. Such outcomes are highly significant, from both moral and legal perspectives, if property or lives are lost [13]–[17].

III. Mission Definition as Goals or Tasks With Runtime Constraints

A. Goal Definition and Task Decomposition

Research by the authors and our colleagues at the Naval Postgraduate School (NPS), Monterey, CA, USA, relating to unmanned maritime vehicle mission definition and control has extended over a period of more than two decades, and has included successful open-ocean testing of two autonomous vehicles [1], [2], [18], [19]. Based on these efforts and prior operational experiences with vehicles afloat, the authors recognized the need to view maritime robot software development

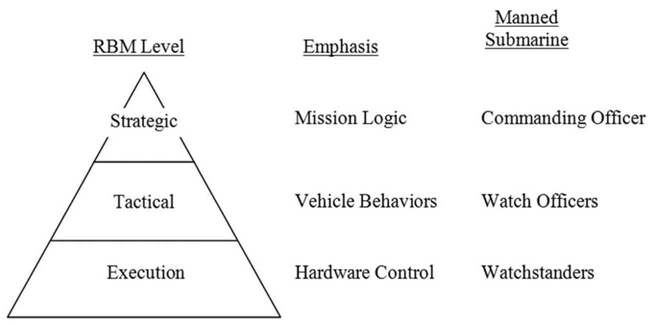


Fig. 1. RBM software architecture is based on hierarchical control paradigm of naval vessels [21].

from both a top-down and a bottom-up perspective, and have incorporated both approaches into a trilevel software architecture called the rational behavior model (RBM) [20], [21]. The RBM architecture involves three layers, each requiring different types of software roles corresponding to human roles. A variety of other three-level robot architectures have been proposed and implemented over the past two decades, typically robot-specific rather than generally repeatable, often with similar timing principles and varying jargon [22]–[24]. RBM is modeled on the command hierarchy of manned ships and aircraft, organizing robot control requirements into execution, tactical, and strategic levels as depicted in Fig. 1.

Specific details regarding the design characteristics of each RBM level follows.

1) *Execution level control* includes those hard-real-time uninterruptable tasks associated with control and management of hardware systems that directly interact with the vehicle's physical environment. These feedback-driven controllers correspond to the activities of a manned vessel's junior crew members and include manipulation of control surfaces and sensors. Typically, most execution-level software is provided by the manufacturer of a given robotic vehicle.
2) *Tactical level tasks* direct execution level functionality to realize more complex behaviors. Task behaviors correspond to management by a manned vessel's watch officers and can be as simple as directing a desired course and speed, or transiting to an ordered geographic location, or conducting specialized tasks such as an area search, mapping, rendezvous, etc.
3) *Strategic level goals* are at the highest level of control and correspond to guidance directed by a manned vessel's commander. These goals control overall mission conduct by triggering tactical level behaviors.

The RBM strategic level is entirely concerned with carrying out mission logic. Any form of mission definition at this level can be formalized as a finite state machine (FSM) in which each node of the machine's state graph corresponds to commanding (calling) a tactical behavior, with subsequent logical branching depending on the value returned by the behavior.

The RBM tactical level carries out mission tasks, typically using a vehicle controller and discrete decision process similar to the depiction in Fig. 2 [25]. With this model, the controller

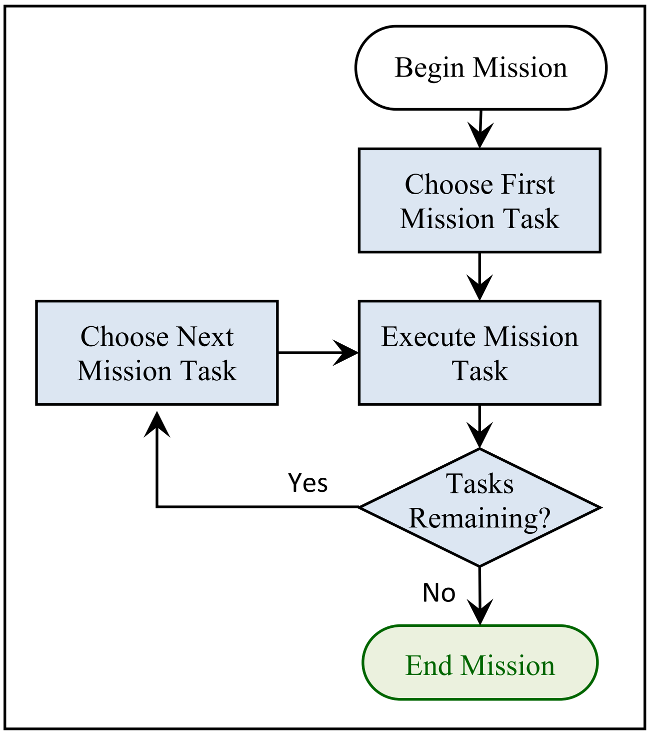


Fig. 2. Strategic level task sequencing algorithm for mission conduct as a series of discrete mission tasks and associated decisions [25].

periodically takes stock of the current situation, determines status of the current task, and proceeds to the next task when the current one is complete. Tactical level RBM software modules are often referred to simply as behaviors [4], [5] and are executed primarily for their "side effects" that accomplish activity by invoking execution level atomic behaviors. That is, behaviors cause the vehicle to interact with its environment, internal or external, to accomplish a specific mission goal or subgoal.

Tactical behaviors return result values from a carefully defined finite set. Original work demonstrated broad tasking flexibility using either Boolean success or failure. Superior mission flow logic is possible using ternary values, i.e., success, failure, or exception. Example exceptions are expressed simply as constraints, for example "task incomplete due to phase timeout" or "task canceled due to (potentially) violating an ethical constraint" [5].

Execution-level processes provide closed-loop and open-control of vehicle effectors and actuators (such as propellers, rudders, actuators, etc.). Execution level commands often match common human commands such as "Come Left to Course North," "All Stop," or "All Engines Ahead Full." In practice, this type of decision process is commonly referred to as a sense–decide–act (SDA) loop when referring to overall control loops for computational autonomous agent activities, or sometimes a sense–interpret–decide–act loop when emphasizing machine evaluation of sensor inputs [26]. Comparable patterns for effective human behavior are characterized as observe–orient–decide–act (OODA) loops [27].

The motivation for adopting RBM in human/robot mission software development is to avoid enigmatic and monolithic

Goal 1: **Proceed to Area A and search the area.** Next, 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.** Next, if the sample is obtained, execute Goal 3. If the sample cannot be obtained, execute Goal 5.

Goal 3: **Proceed to Area B and search the area.** Next, upon either search success or failure, execute Goal 4.

Goal 4: **Proceed to Area C and rendezvous with vehicle 2.** Next, upon rendezvous success or failure, execute Goal 5.

Goal 5: **Proceed to recovery position (mission complete).** Next, upon successful arrival, mission complete. If unable to return to base, abort the mission.

Fig. 3. Plain-language strategic level search and sample mission, providing well-structured success-failure branching for human approval [32]. This mission is also displayed graphically in Fig. 4.

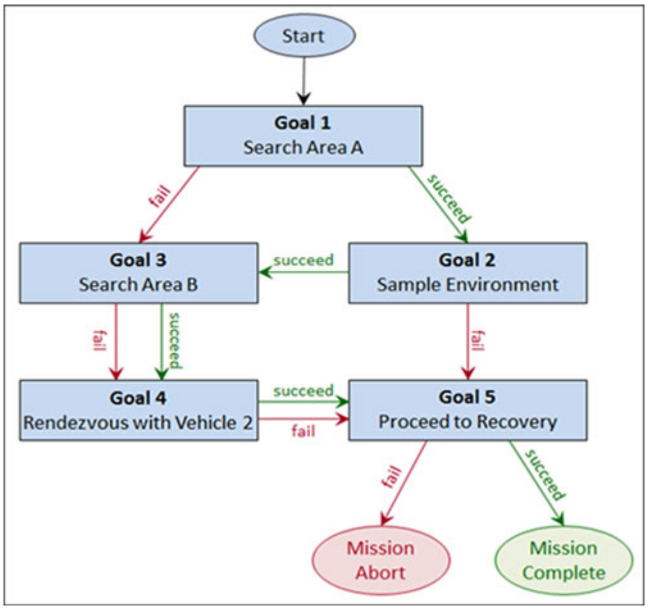


Fig. 4. Binary strategic-level mission flow graph depicting transition logic of exemplar search and sample mission, suitable for human approval of comprehensive mission orders to unmanned systems [33]. Mission corresponds to human-directed goals in Fig. 3.

software, and instead use layered software based on metaphors familiar to human beings in exercising their duties at each level of a mission [21], [20]. Evidently, this kind of approach allows assignment of responsibility to just one human being for any tested mission capability, thereby eliminating “finger pointing” and confusion in case of mission failure due to either conflicted guidance or coding errors. Such structuring enables coherent robot/human mission control software so that necessary levels of human accountability can be assured. Figs. 3 and 4 show a plain-language mission description with matching mission-flow logic, suitable for human supervisory approval and autonomous system tasking.

B. *Premission Testing and Flow Graphs*

In order for every possible path through a strategic level graph to be exhaustively traced by the responsible mission specialist to ensure its correctness, mission-flow tasking must be acyclic and free of sequencing loops. Such a graph can be said to be fully testable in the sense that a knowledgeable human being can certify that every possible sequence of values returned from successive phase execution results in the desired outcome. That is, if loop free, the mission graph can be manually fully validated and can serve as the formal basis for specifying mission control code.

Multiphase missions planned and executed by human beings typically begin with a clearly stated high-level objective. Then, utilizing trusted behaviors executable by subordinate units (or by corresponding software modules), individual mission phases are defined and then connected to form (implicitly or explicitly) a directed graph, such as Fig. 4, often called a process flow graph. If there are no branches in this graph, it is sometimes called a script. In a well-controlled and thoroughly understood environment, a script is often sufficient for human tasking. However, for robot execution, or for humans in an uncertain or hostile environment, phase failure must be explicitly accounted for. This is accomplished by branching based on a predefined finite set of possible phase-execution outcomes. Fig. 4 illustrates specification of a maritime mission in which phase outcomes are binary choices. That is, as can be seen, each phase ends with either success or failure.

It is commonplace in the civilian world to specify complex multiphase tasks by means of binary process flow graphs. It is our belief that such graphs are understood well enough by the population as a whole that they can be used in trials and other legal proceedings to establish ownership of processes and procedures as well as liability in cases of product failure. They are therefore highly suitable for human use in specifying mission orders at a strategic level. Moreover, any acyclic flow graph can be exhaustively tested and documented as a quantitative proof that the specified graph correctly implements the intended mission [5].

C. *MEA: Reinterpreting Mission Flow Graphs*

Since the strategic level of RBM is based on logic, at first it appeared that predicate calculus might be needed in its implementation, and that strategic level mission execution is most properly viewed as theorem proving. Therefore, the Prolog (programming in logic) language [28] was chosen for implementing the strategic level of initial sea trials with the Phoenix AUV [4]. In these experiments, mission phases and branching, defined in Prolog, were activated by querying a top-level mission predicate.

TABLE I
FORMAL MATHEMATICAL DEFINITION OF MISSION EXECUTION AUTOMATON (MEA)

MEA is formally defined as follows:
$M = (Q, \Gamma_i, \Gamma_r, b, \delta, \gamma, q_0, F)$,
where:
Q = a finite, non-empty set of *states*
Γ_i = a non-empty set of input symbols corresponding to *behavior-initiation function calls* to the tactical level (note, once parameterization of function calls is taken into consideration, this set is of potentially infinite size but is practically constrained to a finite set by γ)
Γ_r = a finite, non-empty set of *response symbols* corresponding to *return values* from behavior function calls
$b \in \Gamma_i$ is a *blank* and equates to no function call (note, since no function call is made, no response will be received, so execution will halt in the current state)
$\delta : (Q \ \ F) \times \Gamma_r \rightarrow Q$ is the *transition function* mapping a current state and response to a new state
$\gamma : Q \rightarrow \Gamma_i$ is the *behavior call function* that maps a state to a behavior-initiation function call
$q_0 \in Q$ is the initial state
$F \subseteq Q$ is the set of final states which equate (if the mission is constructed correctly) to mission termination
Constraints can be added explicitly to the definition by adding a set of constraints C,
a constraint-mapping function $\tau : Q \rightarrow c \subseteq C$, and modifying both Γ_i and γ to account for τ.

While testing was successful, and the vehicle returned safely, there was no means of verifying mission correctness since predicate calculus is generally not provable. That is, a set of facts and rules defining a formal system cannot, in general, be shown to be complete and free of contradictions [29]–[32].

In an attempt to achieve mission provability in a second unmanned underwater vehicle (UUV), the Aries [2], rule-based mission definitions were adapted as active state graphs such as described above. That is, while still using Prolog for exhaustive premission testing of mission flow graphs, in-water real-time software treated each mission phase as a software object, with associated methods to accomplish phase actions and transitions. A task sequencer was added to properly cycle through the phases of a mission in response to values returned from calls to phase methods. Such return values, in turn, come from an external agent that might be either a human being or a mechanical system (robot, external memory, etc.). Fig. 2 defines the action of such a sequencer.

In the programming environment for Aries, called "AUV Workbench," [2], [33] a graphical menu system is used to define a mission state graph that was then incorporated with a mission sequencer to produce real-time code for the strategic level. In this case, since Aries was fully autonomous, queries from phases were directed entirely to the tactical level software with Aries itself acting as the sole external agent for the active state graph (FSM) directing the mission.

From an algorithmic perspective, it is noteworthy that "behavior call functions" are the same as the "external agent communication functions" previously defined in [32], [34]. It should also be recognized that, in case the MEA has an external one-dimensional memory (i.e., a bidirectional tape recorder) as its only external agent, then it is exactly a Turing machine (TM) [30], [35], [32]. This is important because, as is well known in computer science, a TM can return a value for any computable function. This means that any alternative means for implementing the strategic level of the RBM architecture cannot be more computationally expressive or powerful than an MEA. This is an important and previously unreported finding of the present paper. Table I provides a formal definition of the MEA.

Based on formal principles of computational theory, it is further possible to define and implement a universal TM (UTM) in which the state table for a specific machine is stored on the "tape" of the UTM [30]. Likewise, it is possible to transform any given MEA into a mission execution engine (MEE) with a corresponding file holding mission orders. Such an MEE, along with mission orders for the mission of Figs. 3 and 4, has been mathematically defined in executable predicate calculus form using the Prolog programming language [28], [32]. Interpretation and execution of this mission by the MEE provide a mathematically rigorous basis for comparable implementations using any other computer language.

D. Progressive Goal Refinement

In describing complex tasks to subordinates, humans often subdivide these tasks into a series of subordinate tasks that can be executed to accomplish the overall mission. For instance, a complex task (or mission) during which a manned vehicle is expected to conduct searches and collect environmental samples before rendezvousing with another manned (or unmanned) vehicle might be specified as a series of tasks as depicted in Fig. 4. Providing the vehicle's operator knows the geographic characteristics of areas A, B, and C and understands what the commander means in directing searches, environmental sampling, and rendezvous, the operator is able to reliably execute this mission as specified.

Note that each of the above tasks is nontrivial. Most tasks include transit as well as subsequent operations in different locations. Each task requires multiple sophisticated steps for successful completion, whether accomplished by a human or a robot. Each subtask typically requires even more specialized capabilities. For example transit requires safe navigation, which requires sensing and classification for situational awareness plus stable control, which in turn requires operation of hardware/software capabilities, and so on. Each level of abstraction requires different capabilities and sophistication, while no layer of capability can exist correctly without the corresponding layers of functionality that lie above and below. Thus, task decomposability is essential.

It is evident that an MEA can actually carry out the mission it defines only when it is properly structured for a given vehicle and mission. One obvious requirement is that all phases in a given

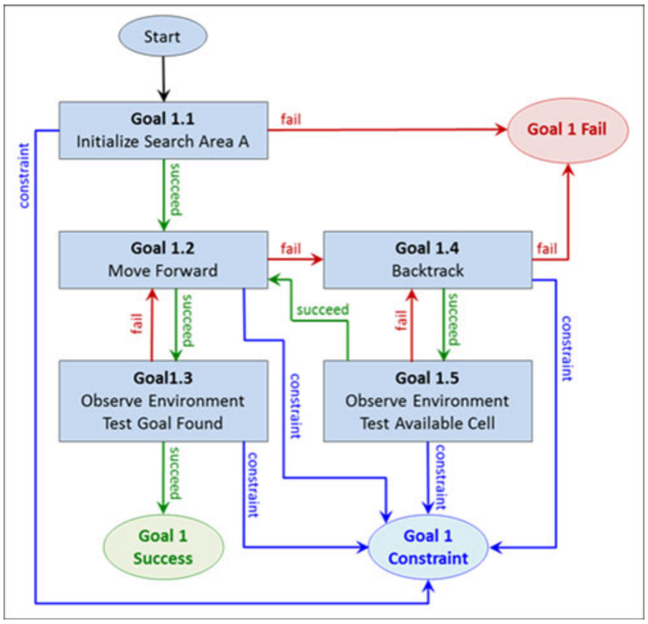


Fig. 5. Progressive refinement, illustrating the internal flow graph for grid-based depth-first search of area A, corresponding to necessary subgoals within goal 1 of Fig. 4, adapted from [35].

MEA graph must correspond to an available top-level trusted behavior. Consider, for example the case that a general open-ocean area search trusted behavior is available (such as a box search or spiral search [1]), but that area A, referred to in Fig. 4, is in fact a coastal area known to be full of unmapped rocks and shoals. In such a case, an entirely different type of search is called for. In particular, if the vehicle always knows where it is in global coordinates (through GPS, inertial navigation, etc.), depth-first search, widely utilized in AI applications, can be used [36]. Fig. 5 below illustrates one implementation of depth-first search appropriate to grid-based terrain exploration.

Superficially, Fig. 5 resembles Fig. 4 of this paper. However, thoughtful analysis shows several important differences. First of all, Fig. 5 contains several loops. This means it cannot be exhaustively tested except for a given terrain example with a finite number of cells. Even then, possible user response sequences would likely be so long and so many in number as to make exhaustive testing infeasible. Further, such a test would be valid only for the given terrain. Thus, the depth-first search algorithm of Fig. 5 is not exhaustively testable and thus is not suitable for use as a top-level behavior in an overall mission graph. Moreover, backtracking requires either physical terrain marking (generally undesirable or infeasible in military operations) or construction of a map (implying an external memory). In case a map is used, this requires external storage, and the MEA then becomes a variety of TM. However, it is generally known that the correctness of a given TM might not be provable. What can be done about this?

Fortunately, depth-first search as presented here mechanizes a classic trail-blazing method used by human explorers in seeking a goal. This method is known to always succeed when the number of cells to be searched is finite. Thus, it can be used as a trusted behavior, though not at a top level, because the time required to search an arbitrary terrain cannot be predicted. This shortcoming can be overcome by a slight modification to Fig. 5 that causes it to return a value of failure (i.e., exception) if a specified time duration (since search commencement) is reached. Thus, the inability of depth-first search to complete due to a time-out failure is not as serious as it might seem at first. Specifically, as can be seen from Fig. 4, the overall mission plan makes provision for such a “giving up too soon” failure, and simply sequences the next goal which is to search area B.

E. Classical Decision Logic for Task Sequencing

It is noteworthy that similar SDA or OODA control-loop models can be applied to both human and nonhuman operators. Such correspondences imply that missions thus specified might be executable not only by humans, but by human-controlled robots, human–robot teams, and carefully constrained autonomous robots as well. However, one important aspect of the mission above must be accounted for. Simple sequential execution of tasks as in a mission script implicitly assumes success for each task. Where human operators are concerned, this is acceptable in most circumstances. When the ability to complete a task is in question, a human operator is able to request guidance from higher authority or use his best judgment to decide how to proceed. Under the requirements underpinning the framework proposed in this paper, this is not necessarily an option for robot agents. Rather, the course of action that the vehicle is to undertake in the event of task failure must be fully specified in the mission description. This can be achieved through the introduction of a simple branching structure.

As discussed previously, a specific autonomous agent may be trusted to execute a finite set of atomic behaviors that are used to define the mission. Further, the agent must be capable of detecting when a behavior is successfully completed and when the behavior cannot be successfully completed. It follows that a vehicle must be able to detect the success or failure of tasks within the mission definition so long as those tasks are comprised of trusted behaviors. This capability makes it possible to more rigorously define missions in a way that target autonomous vehicles can be trusted to execute without direct supervision.

With the introduction of potential branching based on task success or failure, overall mission success is no longer reliant on a fixed sequence of task executions. In fact, a particular mission can include the successful completion of some tasks, the failure of different tasks, and the complete omission of others. It is appropriate in this context to refer to the individual tasks as goals to be achieved rather than simply as tasks. Interestingly, the SDA/OODA decision loop of Fig. 2 is still suitable for controlling the execution of this revised mission.

The binary-branching flow graph in Fig. 4 is one among many potential representational forms for this and many other missions, and a number of graphical, programmatic, and XML-based definition forms have been proposed [1]–[3]. This flow-graph encoding is of particular interest because it provides an intuitive depiction of a potentially complex mission. In fact, an operator or supervisor can utilize a mission specification of this form to mentally “rehearse” the mission by intentionally

Constraint 1:	The vehicle must maintain navigational accuracy within acceptable limits. Applies to entire mission.
Constraint 2:	All safety equipment must be fully functional. Applies to entire mission.
Constraint 3:	All mission systems must be operational. Applies to Goal 1, Goal 2, and Goal 3.
Constraint 4:	Acceptable distance from shipping lanes in the form of 1000 meter lateral standoff or minimum depth of 20 meters must be maintained. Applies to Goal 1, Goal 2, Goal 3, and Goal 4.
Constraint 5:	Must be able to detect surface contacts within 5000 meters. Applies to entire mission.
Constraint 6:	Detected surface contacts are to be avoided by a minimum of 1000 meters. Applies to Goal 1, Goal 2, Goal 3, and Goal 4.
Constraint 7:	Minimum depth of 20 meters is to be maintained. Applies to Goal 5.

Fig. 6. Constraints suitable for careful prior human supervision of robot actions in the exemplar search and sample mission.

traversing the graph from start to finish while exhaustively testing success and failure branches at every step. While not yet providing the required level of mathematical rigor, this ability to informally traverse all possible task sequences in this manner is an important step toward providing assurance to the responsible operator that the mission will progress according to human intent under all foreseeable circumstances.

F. Adding Constraints to Mission Decision Logic

As presented so far, this mission definition paradigm does not explicitly address the issue of ethical mission execution. Specifically, no mechanism has been suggested at this stage to define ethical constraints affecting the overall mission or individual tasks. It might be casually argued that ethical conduct is implied by "successful" completion of goal's requirements. However, such an assumption is naïve and does not provide nearly enough confidence for the operator to assume liability for the mission's conduct. For instance, it is apparent that an UUV with an appropriate search behavior can achieve goals 1 and 3 of the example mission. Unfortunately, it may or may not be able to do so while avoiding detection, remaining clear of other vehicles in the area, or maintaining a specific navigational accuracy. If any of these (or other) conditional requirements must be met in order for the goal to be achieved in a safe and ethical manner, then an additional mechanism must be provided to incorporate those ethical constraints into the mission specification.

Common approaches to ethical oversight presume the existence of some oracular agent equipped with morality and philosophical knowledge. We specifically reject such notions as ill-defined, not implementable, untestable, and an abdication of necessary human responsibility. A more practical approach is necessary that keeps human operators "in charge" of robot actions, even when operating at a distance with gapped communications [5], [16], [15].

Constraints on mission tasks are far more precise and effective, matching common human practices. In a mission context, ethical constraints do not describe characteristics of individual goals, but rather what must be considered and enforced during goal execution. From the standpoint of operator accountability, the constraints must be specified in a manner that preserves the ability to trace high-level mission flow, and also specified in a way that can ultimately be monitorable and enforceable by the autonomous vehicles themselves. A plain-language version of legitimate constraints is given in Fig. 6.

Ethical constraints vary and may be intuitively applied to either an entire mission or to relevant individual goals as appropriate. That is, there may be certain constraints that must be enforced from launch until recovery (e.g., all safety systems must remain operational), and others that only need to be enforced during the execution of specific goals (e.g., maintaining safety depth in the search area).

Up to this point, the definition scheme only provides for binary branching of the mission-flow diagram: Once initiated, a goal either succeeds or fails, and the mission then proceeds accordingly. Such a representation is fully representative of any decision tree, since tree graphs of arbitrary branching size can be traversed in a binary manner. However, a binary approach also presumes that an impending ethical constraint violation equates to goal failure. Such equivalence might be acceptable in many cases, and ethical violations causing goal failure certainly result in correct application of the constraints in the sense that goal execution no longer proceeds in the face of constraint violations. On the other hand, it might well be desirable to treat responses to impending constraint violations differently than simple failure (for example, to meet an additional independent objective). A more-responsive approach is possible through the addition of a third potential goal-execution outcome for constraint violations, along with a corresponding branching option in the mission flow structure. That is, execution of an individual goal becomes terminated upon goal success, goal failure, or impending violation of a constraint applied to that goal. Flow of control then proceeds as directed to whichever subsequent goal is next designated as appropriate.

The question before us now is to find a way of modifying the binary-branching approach of Fig. 4 to include these limitations on vehicle behavior into the mission state graph. An excellent solution turns out to be remarkably straightforward. All that is required is to allow ternary branching in this graph with the third branch applying to situations when a constraint is about to be violated. This constraint-based tree approach is shown in Fig. 7. It is quite useful and an excellent match for supervisory planning needed when humans perform robot mission planning. The general expressive power of binary-flow logic is

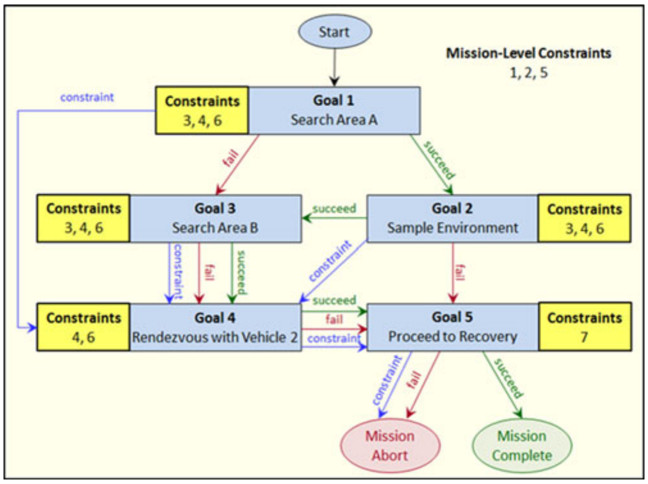


Fig. 7. Mission-flow graph for a search and sample mission with ternary branching for imminent ethical-constraint violations. Constraint definitions are provided in Fig. 6.

preserved, and system responses must be explicitly considered for each success, failure, or constraint violation that might occur when performing each mission goal. The ternary-flow structure is also similar to exception handling in modern programming languages, which can facilitate implementation and testing.

G. Implications of Ethical Constraints on Mission Tasking

Designing robot missions in the form of a flow diagram consisting of a set of discrete goals, with ethical constraints applied to individual goals as described here, provides an intuitive mechanism that can enhance responsible operators' understanding of the missions they expect to supervise. The nature of the mission specification is declarative. At this level of abstraction, individual goals execute sequentially according to the mission graph, irrespective of elapsed time, and each goal predictably terminates in one of three possible states (goal success, goal failure, or constraint violation).

Supervisory trust that a directed vehicle can execute specific goals, recognize goal failure, and identify pending constraint violations provides important boundaries on autonomous behavior. Essentially this approach eliminates any need to make assumptions or guesses concerning intended vehicle conduct during goal execution. Rather, the necessary requirement of well-specified tasking is specifically placed on human operators to create well-defined and thorough missions. Further, if the size of the mission-flow diagram is reasonably managed, then exhaustive testing of all possible mission execution sequences is achievable and tractable. These aspects of mission design are fundamentally important, and are essentially quite similar to the essence of coordinated operational tasking among ships and aircraft led by responsible and cooperating humans. Section III-H strengthens the foundations for these concepts. Examining the underlying nature of the mathematical formalizations used here can provide further operator assurance that a particular mission is appropriately defined and can proceed as expected, in an appropriate matter, under all circumstances.

H. Summary of Insights, MEA

As a generalization of the TM, the MEA provides a mathematically sound approach to the definition and exhaustive testing of unmanned vehicle missions. The MEA includes a mission-specific FSM and unlimited memory. Consistent with the MEA generalization, the TM tape can be replaced by a physical robot or a human being to which output can be sent (commands) and inputs can be received (e.g., success, failure, or constraint-violation responses). Other external agents can be optionally added as well.

To guarantee eventual termination of a mission, the structure of a strategic level mission must be constrained somewhat beyond the basic MEA definition. Specifically, the mission FSM cannot include loops, unreachable states, or sink states (i.e., non-terminal states from which further transitions are not possible). Further, the strategic level mission must be defined with few enough states and transitions to allow for tractable exhaustive testing by a human operator. However, when a strategic level goal is iteratively refined to develop a tactical level behavior (as with the depth-first search example) these restrictions do not apply since the tactical level can implement a timeout failure to ensure termination of individual behaviors.

Finally, a universal MEA can be achieved by implementing sequencing and communication functions as a separate MEE and then developing the mission flow graph as a set of mission orders in a form understandable by both the MEE and humans who are mission specialists, but who may not be programmers. There are many choices for expressing such mission orders including flow charts, text-based programming languages, and graphical user interface techniques.

IV. Validation of RBM Software Architecture Through Real-World and Virtual Experimentation

A. Testing Control-Logic Responses

Frequently applying the double-check question "how might a human accomplish this task?" is an important design principle for autonomous-system mission production. Simulation testing can help demonstrate mission clarity and expected responses at all levels of sophistication. Fig. 8 shows an example human-directed simulation trace of RBM control logic. Of further interest is that both mission tasking and supervisor inquiry/response are similar to the qualification testing required of human operators performing similar roles.

A similarly capable, independent implementation uses the Hierarchical Task Network behavior model and Python programming language in the Combined Arms Analysis Tool for the 21st Century (COMBATXXI), a simulation tool developed and used by the U.S. Army and U.S. Marine Corps within various analytic studies [37], [38].

Long-duration robot missions may have lengthy mission orders. The ability to test missions exhaustively through manual or automatic means, together with the ability to isolate faults, is an important design requirement. Testability provides a significant boost to improving human confidence that robots can correctly perform assigned tasks in an appropriate

```
Ethical Mission Execution Trace #1:
?- execute_mission.
Commence: Search Area A.
Did goal Succeed (s), Fail (f), or end with a Constraint (c)? constraint.
Commence: Rendezvous with vehicle 2 in Area C.
Did goal Succeed (s), Fail (f), or end with a Constraint (c)? constraint.
Commence: Return to Base.
Did goal Succeed (s), Fail (f), or end with a Constraint (c)? constraint.
Mission Abort!

Ethical Mission Execution Trace #2:
?- execute_mission.
Commence: Search Area A.
Did goal Succeed (s), Fail (f), or end with a Constraint (c)? succeed.
Commence: Take environmental sample from Area A.
Did goal Succeed (s), Fail (f), or end with a Constraint (c)? succeed.
Commence: Search Area B.
Did goal Succeed (s), Fail (f), or end with a Constraint (c)? succeed.
Commence: Rendezvous with vehicle 2 in Area C.
Did goal Succeed (s), Fail (f), or end with a Constraint (c)? succeed.
Commence: Return to Base.
Did goal Succeed (s), Fail (f), or end with a Constraint (c)? succeed.
Mission Complete!
```

Fig. 8. Typical execution traces from exhaustive premission strategic level testing of the exemplar search mission with ethical constraints depicted in Fig. 7, tested using MEA Prolog source code (bold font indicates operator input).

order, and also correctly avoid prohibited situations forbidden by constraints. Testability also enables eventual certification of otherwise-diverse robot systems as "qualified" to follow human direction competently, similar to certification requirements for any other human-directed vehicle.

B. Mission Representations and Syntactic Validation

Up to this point, all results presented have related to the strategic level and the tactical level of RBM software for a single example of a "search and sample" mission for a notional autonomous underwater vehicle. Furthermore, all results presented thus far have been obtained from high-level mission simulations typically written in the Prolog logic-programming language. However, beginning in 1993, in parallel with formalization and publication of details of RBM [20], [21], the authors and their collaborators demonstrated the value and practicality of this approach for undersea robots through a series of open-ocean experiments involving two small unmanned submarines. Alternatives have been implemented and tested using Allegro Common Lisp and the C language integrated production system (CLIPS) rule-based expert system [39], thus further demonstrating functional correctness whether using forward chaining or backwards chaining to resolve task logic. These experiments and results obtained are summarized in the following sections.

Aries AUV missions were defined with AVCL, a schema-constrained XML data model supporting autonomous vehicle mission definition, execution, and management [1]–[4]. While the mathematical concept of an MEA had not been developed at the time of AVCL's development, AVCL does provide a fixed set of goal types including area search, environmental sampling, and rendezvous and is thus suitable for the definition of mission flow diagrams such as the one depicted in Figs. 3 and 4. Further, AVCL was intentionally designed to support implementation of the RBM strategic and tactical levels and was utilized to define RBM-controlled Aries missions for simulation in the AUV Workbench virtual environment and for open-ocean real-world tests.

```
<Goal description="search operating area A" id="Goal1" >
    <Search datumType="area" requiredPD="0.8" singleTarget="false" />
    <OperatingArea>
        <Rectangle>
            <NorthwestCorner>
                <LatLonPosition latitude="36.69" y="-121.90" />
            </NorthwestCorner>
            <Width value="500.0" />
            <Height value="3000.0" />
        </Rectangle>
        <DepthBlock minimum="25" maximum="75" />
    </OperatingArea>
</Goal>
```

Fig. 9. XML-based AVCL specification of Goal 1 from Fig. 7 for execution on the NPS Aries UUV [1].

As an example, consider the XML snippet of Fig. 9, which provides a hypothetical description of goal 1 from Figs. 3 and 4 for execution by a UUV. This specification defines the type of search to be conducted (area search for multiple targets with an expected probability of detection of 0.8), the area to be searched (a 500 by 3000-m rectangular area with a northwest corner at 36.7 north latitude and 121.9 west longitude), and stipulates that the search be conducted at a depth of between 25 and 50 m. Evidently, the search goal definition describes what is required for successful completion of the goal. However, it does not dictate precisely how the goal is to be completed since such navigation and maneuvering decisions remain the responsibility of the tactical-level implementation.

Simulation of a mission consisting of an AVCL specification for a search goal similar to the one in Fig. 9 and avoid areas specified as constraints in the AUV Workbench is shown in Fig. 10. During the mission, the tactical level plans a path and

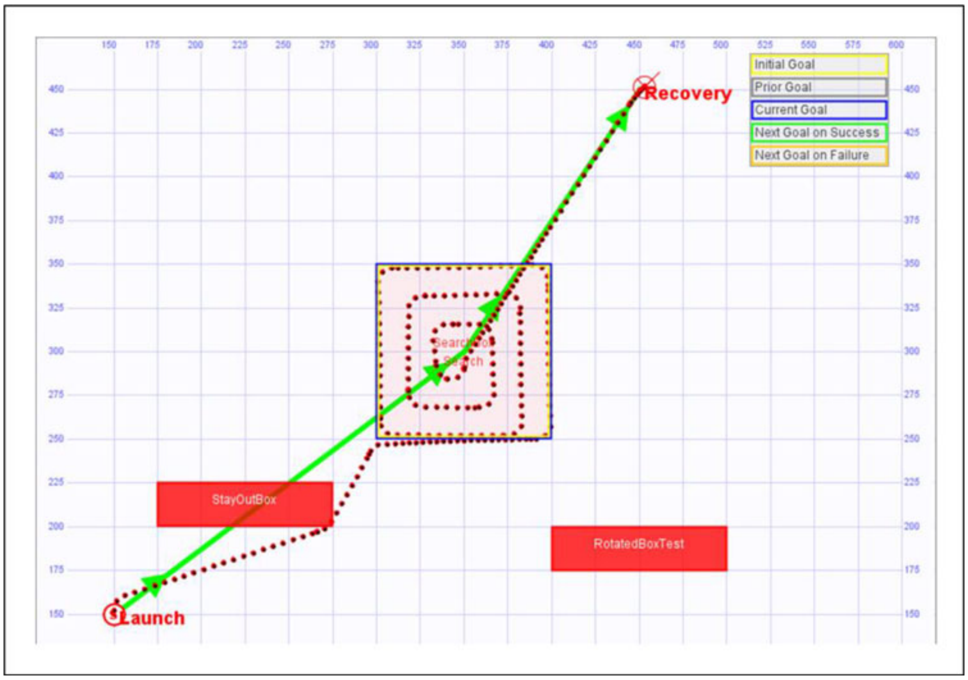


Fig. 10. AVCL mission SimpleBoxTest.xml demonstrating simulated conduct of a goal-oriented mission that was performed amidst constraints [19]. Simulation replay rendering produced by AUV Workbench [33].

maneuvers to the search area while remaining clear of the avoid areas and then develops and executes a suitable pattern for the required area search. More complicated missions demonstrating the binary branching model were conducted in AUV Workbench simulations and also in open-ocean experiments in Monterey Bay [1], [33]. A comprehensive comparison and consolidation of diverse goal types can be found in [1].

C. Dangers Associated With Using Rules and Fact Assertion to Implement Strategic Level Logic

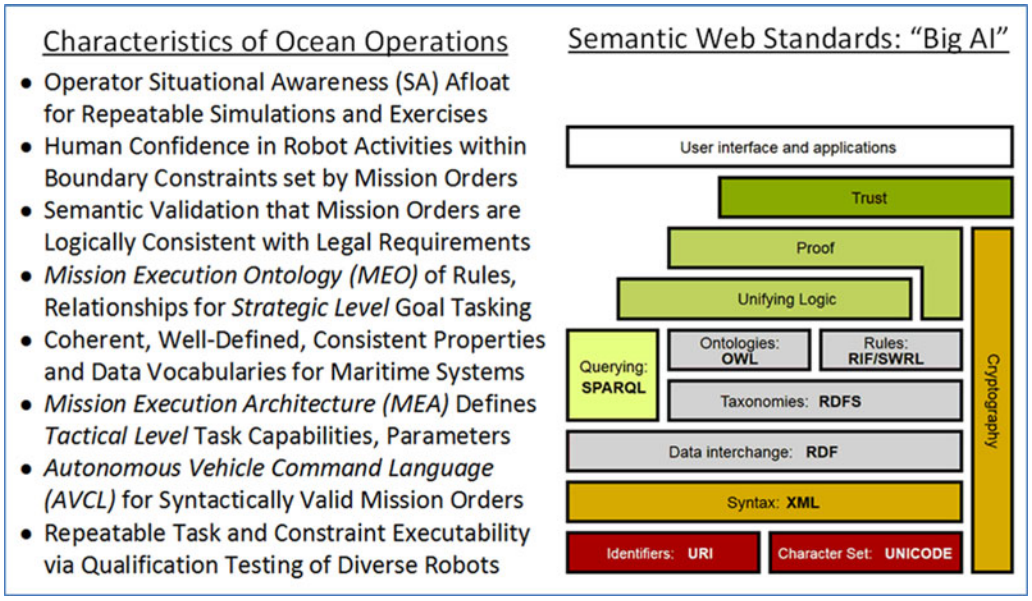
As described above, Phoenix missions were executed as a result of an inferencing and reasoning process, using a set of rules and facts for mission definition. While all in-water missions succeeded, and Phoenix was never lost at sea, this approach provided no means of proving the correctness of strategic level software comparable to the exhaustive testing made possible by the MEA formalism. The authors believe that this is a serious limitation that applies to all approaches for top-level strategic level mission definition that require specific actions to be derived from general principles rather than using a completely concrete FSM approach.

Specifically, a set of rules and facts amounts to a formal mathematical system in which the rules and facts serve as axioms. Theoretically it is known that, in general, no such set of axioms can be proved complete. Here, completeness means that all true theorems can be proved by formal application of predicate calculus. Such computability properties are hard to prove. In fact, to the astonishment of the entire mathematical world, Gödel proved in 1931 that such a simple system as integer arithmetic cannot have any axiomatic basis. Perhaps equally shocking, even plane geometry had no sound axiomatic basis until around 50 years ago. This meant that, from a strictly formal perspective, all of Euclid's original "proofs" were merely plausibility arguments. Fortunately, all of the theorems believed to be true are in fact provable using the complete and consistent set of modern algorithms [31].

The significance of the above observations relative to top level mission specification derives from the fact that, for rule-based systems, mission execution can sometimes be regarded as a side effect of proving the theorem that "there exists a way to satisfy all specified mission goals while observing all given constraints." If it eventually turns out that the mission axiom set contains a contradiction, then system execution behavior becomes unpredictable and not testable, as well as potentially hazardous and even self-defeating.

The potential unpredictability of less-formal reasoning approaches and the inability to prove the correctness and completeness of axiomatically defined missions effectively precludes formal responsibility or liability for robot missions using these approaches. In fact, AI approaches to top-level mission specification and control almost invariably make use of some form of reasoning and/or statistical pattern recognition. Applying such broad abstractions to the innumerable situations that can arise in the real world is very dangerous when applied to potentially lethal robots, and also makes the assumption of responsibility by human operators unrealistic. It is therefore apparent that the abstract reasoning of general AI approaches is inappropriate at the highest level of robot mission definition and control.

Algorithms cannot replace human responsibility. Even so, a fully testable technology such as that provided by the MEA formalism, allows for assignment of human accountability when directing outcomes and alternatives for robot missions. It is possible that ever-emerging AI techniques may someday provide good methods for achieving specific individual tactical level behavior modules. Such employment of AI capabilities (even when



Characteristics of Ocean Operations

- Operator Situational Awareness (SA) Afloat for Repeatable Simulations and Exercises
- Human Confidence in Robot Activities within Boundary Constraints set by Mission Orders
- Semantic Validation that Mission Orders are Logically Consistent with Legal Requirements
- *Mission Execution Ontology (MEO)* of Rules, Relationships for *Strategic Level* Goal Tasking
- Coherent, Well-Defined, Consistent Properties and Data Vocabularies for Maritime Systems
- *Mission Execution Architecture (MEA)* Defines *Tactical Level* Task Capabilities, Parameters
- *Autonomous Vehicle Command Language (AVCL)* for Syntactically Valid Mission Orders
- Repeatable Task and Constraint Executability via Qualification Testing of Diverse Robots

Fig. 11. MEO characteristics applied using Semantic Web standards to support ethical operations by human–robot teams.

experimental) can be considered appropriate in these cases since success, failure, and constraint violation remain fully accounted for by the strategic level MEA.

V. Ethical Validation of Mission Definitions

A. Description Logics

Thus far, the discussion of MEA mathematical underpinnings, capabilities, and implementations has focused on providing robot operators the ability to rigorously define and test strategic level missions to ensure high-level mission-flow understanding sufficient for the assignment of accountability for vehicle conduct throughout the mission. However, the ability of an actual target vehicle to execute missions defined in this manner without further translation into a vehicle-specific form has not been addressed. Mathematical logic provides a mechanism for bridging strategic level missions described here and vehicle-specific code for specifying and ordering tactical level behaviors. If properly implemented, formal logic can mathematically enforce MEA semantics in the definition of missions and during execution of those missions on target vehicles.

DLs are a mathematical family of logic-based knowledge representation systems that are used to describe concepts and roles within a knowledge-based system through a set of well-defined operations. DL ontologies can be used to describe the requirements and relationships of a system in a semantically meaningful way. That is, they define not only what the relationships are, but how they operate, how they are to be used, and to what specific entities they apply. DLs provide expressive power almost equal to that of first-order logic. Further, these language constructs have been carefully defined to enable (and indeed guarantee) computationally efficient reasoning that can always identify the existence of hidden relationships and errors in the form of rule violations or contradictions [40]. These are strong capabilities with great potential value.

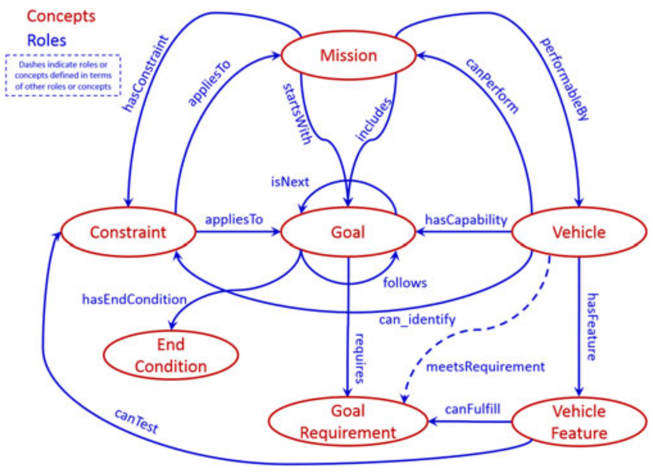


Fig. 12. MEO expressing concepts and roles (i.e., relationships among concepts) representing the flow-diagram MEA mission descriptions.

DLs provide the mathematical foundation of what has come to be known as the Semantic Web, an extension of the World Wide Web [41]. The growth of the Semantic Web has fostered the development of tools and standards that take advantage of DL logical expressiveness and mathematical rigor to provide extensive knowledge representation, discovery, and utilization capabilities. Most notably, the Web Ontology Language (OWL) [42] together with the Resource Description Framework (RDF) [43] encode a particularly powerful DL in a plain-text, XML-based, computer-readable form [44]. Because of its formal and general DL implementation, OWL is potentially useful beyond the Semantic Web domain. It is used here to define a robot mission description and execution ontology that applies and enforces MEA semantics. Further references of interest include [45], [46]. Overall correspondences between ethical characteristics implementable using Semantic Web standards is shown in Fig. 11.

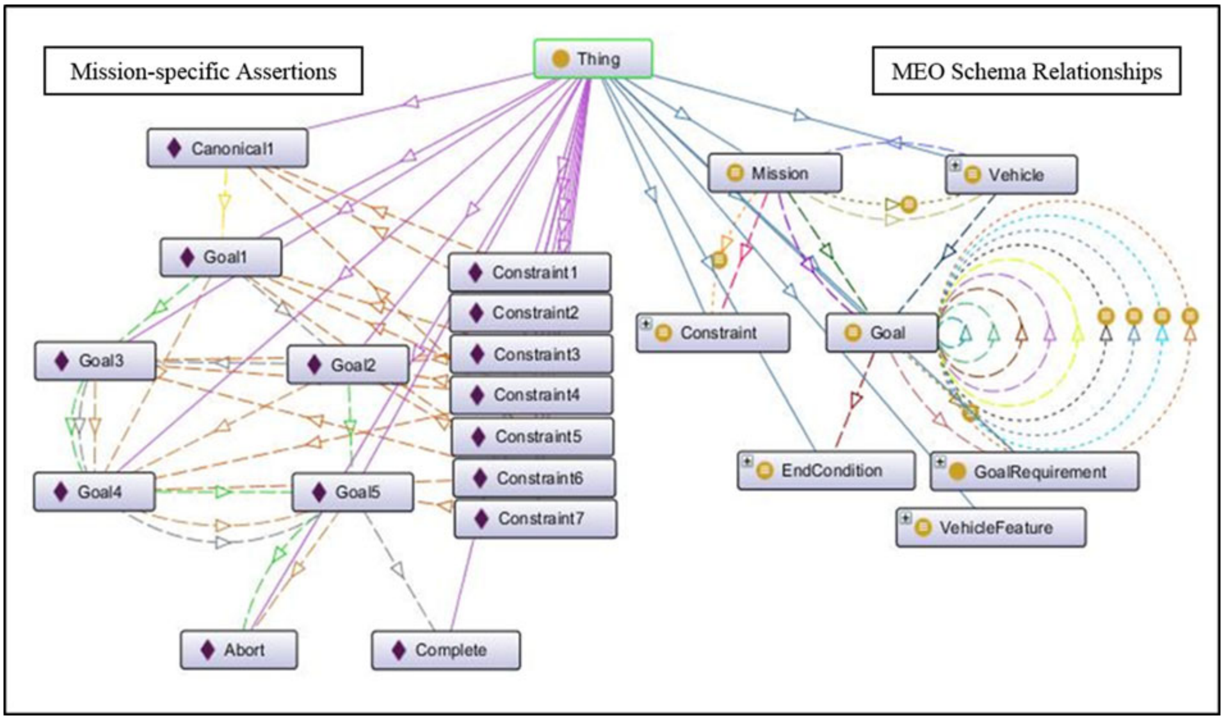


Fig. 13. Validatable RDF/OWL diagram of goals, relationships, assertions, and ethical constraints for the canonical mission of Fig. 7 plus the MEO relationships of Fig. 12. Rules and rendering produced using Protégé Ontology Editor [49], [48].

B. MEO for Robots

The MEO serves a number of purposes. First, it provides a formal and semantically rich description of the characteristics of a MEA mission description. For instance, OWL expressions are used to declare the existence of concepts such as `Mission`, `Goal`, and `Constraint`. OWL statements are also used to define possible relationships (roles) between concepts. An entity to which the `Mission` concept applies, for example, can have an `includes` relationship with an entity to which the `Goal` concept applies. Additional OWL statements describe rules that govern how relationships are applied. As an example, a `Mission` entity must have an `includes` relationship with at least one `Goal` entity and must have a `startsWith` relationship with exactly one of those entities. A graphical depiction of the concepts and relationships defined in the MEO is provided in Fig. 12. As the diagram indicates, concepts and relationships are defined to accurately represent the semantics of the previously discussed flow diagrams to include the definition of individual mission goals and constraints; goal successors in the event of goal success, failure, or constraint termination; the mission's first goal; and the application of constraints either to individual goals or throughout the entire mission.

In addition to the `Mission`, `Goal`, and `Constraint` concepts that are abstracted directly from strategic level mission-flow diagram semantics, the MEO introduces the `Vehicle` concept. This concept provides the ability to include specific target vehicles in the mission-planning process. In particular, the `canExecute` and `canIdentify` relationships allow mission planners to explicitly assert that the intended target vehicle has a tactical level behavior capable of completing a particular goal and recognizing potential violation of a particular constraint, respectively. Evidently, if a mission includes goals for which the `canExecute` relationship does not exist with the intended vehicle or constraints for which the `canIdentify` relationship does not exist, then that mission is not appropriate for that particular vehicle. Given this requirement (which is enforced by rules within the ontology), it is impossible to define a valid mission that cannot be executed by the intended vehicle.

A second important characteristic of a DL-defined ontology is that it not only describes the rules and relationships of a knowledge domain, but also applies those rules and relationships to entities within that domain. Stated differently, the MEO does more than describe what the `Mission`, `Goal`, `Constraint`, and `Vehicle` concepts are and how they relate to one another. It also allows the application of those concepts to real-world entities and the establishment of relationships among those entities. From a practical standpoint, this means that the atomic entities to which the `Goal` and `Constraint` concepts are applied become actual executable specifications for a set of target vehicles.

C. Implementing the Ontology

OWL provides for the incorporation of atomic entities into an ontology using uniform resource identifier (URI) labels that uniquely identify individual entities [47]. Thus, the MEO can be applied to the XML snippet above by defining an OWL statement declaring its existence and corresponding identifier. OWL statements are also used to declaratively apply concepts to and establish relationships between atomic or composite entities within the knowledge base. Fig. 13 illustrates both the example constraint-based mission of Fig. 7 plus the MEO relationships of Fig. 12, as validated and then rendered in the Stanford Protégé ontology development tool [48], [49]. All re-

TABLE II
LOGICAL EXPRESSIONS DEFINING CONCEPTS AND ROLES FOR MISSION PLANNING ONTOLOGY (MEO)

Rules	DL Equations	Plain-language description
M = Mission Rules		
M1	Mission $\sqsubseteq$ ∃startsWith.Goal $\sqcap$ = 1.startsWith	A Mission can only start with a Goal and must start with exactly one Goal
M2	Mission $\sqsubseteq$ ∃includes.Goal $\sqcap$ ≥1.includes	A Mission can only include Goals and must include one of more Goals
M3	Mission $\sqsubseteq$ ∃hasConstraint.Constraint $\sqcap$ ≥0.hasConstraint	A Mission can be constrained only by Constraints and can have 0 or more
M4	startsWith $\sqsubseteq$ includes	A Mission must include the Goal that it starts with
M5	Mission $\sqsubseteq$ ∃performableBy.Vehicle $\sqcap$ ≥0.performableBy	A Mission can only be performed by a Vehicle and can be performable by 0 or more Vehicles
M6	performableBy(M,V) $\sqsubseteq$ ∀(hasConstraint(M,C) ∘ canIdentify(V,C))	A Mission cannot be performable by a Vehicle unless that Vehicle has the ability to identify all Constraints associated with that mission
M7	performableBy(M,V)$\sqsubseteq$ ∀(includes(M,G) ∘ hasCapability(V,G))	A Mission cannot be performable by a Vehicle unless that Vehicle has the capability to accomplish all Goals included in that Mission
V = Vehicle Rules		
V1	Vehicle $\sqsubseteq$ ∃hasFeature.Vehicle_Feature $\sqcap$ ≥0.hasFeature	The only allowable features of a Vehicle are VehicleFeature. A Vehicle can have 0 or more VehicleFeatures
V2	canPerform ≡ performableBy^{-}	performableBy and canPerform are inversely equivalent
V3	meetsRequirement ≡ hasFeature ∘ canFulfill	A Vehicle meets a GoalRequirement if and only if it has a VehicleFeature that can fullfill that GoalRequirement
V4	hasFeature ∘ canTest $\sqsubseteq$ canIdentify	If a Vehicle has a VehicleFeature that can test a Constraint, then that Vehicle can identify that constraint
V5	hasCapability(V,G) $\sqsubseteq$ ∀(requires(G,R) $\sqcap$ meetsRequirement(V,R))	If a Vehicle meets all GoalRequirements for a specific Goal, then that vehicle has the capability for that Goal
F = Feature Rules		
F1	VehicleFeature $\sqsubseteq$ ∃canFulfill.GoalRequirement $\sqcap$ ≥0.canFulfill	A VehicleFeature can only fulfill GoalRequirements and may be able to fulfill 0 or more GoalRequirements
F2	VehicleFeature $\sqsubseteq$ ∃can_test.Constraint $\sqcap$ ≥0.can_test	A VehicleFeature can only test Constraints and may be able to test 0 or more Constraints
C = Constraint Rules		
C1	Constraint $\sqsubseteq$ ∃appliesTo.(Mission $\sqcup$ Goal)	A Constraint can apply to a Mission or a Goal (and nothing else)
C2	Constraint $\sqsubseteq$ ≥1.appliesTo.Goal	A Constraint must apply to at least one Goal
C3	appliesTo ∘ includes $\sqsubseteq$ appliesTo	A Constraint that applies to a Mission must also apply to all of the Goals that Mission includes
EC = End Condition Rules		
EC1	EndCondition ≡ {SUCCEED, FAIL, VIOLATE}	Possible ending conditions are SUCCEED, FAIL, and VIOLATE (i.e., imminent Constraint violation)
G = Goal Rules		
G1	Goal $\sqsubseteq$ ∃requires.GoalRequirement $\sqcap$ ≥0.requires	A Goal can only require a GoalRequirement and may require 0 or more Goal Requirements
G2	Goal $\sqsubseteq$ ∃hasEndCondition.EndCondition $\sqcap$ ≤1.hasEndCondition.End_Condition	A Goal's ending state must be an EndCondition, and a Goal can end with at most one EndCondition
G3	Goal $\sqsubseteq$ ∃isNext.Goal	A Goal can only have other Goals next
G4	hasEndCondition(G,SUCCEED) $\sqcup$ hasEndCondition (G,FAIL) $\sqcup$ hasEndCondition (G,VIOLATE) $\sqsubseteq$ isNext(G,G2)	A Goal can only have an immediate successor based on the existence of an ending state for that Goal
G5	Goal(G) $\sqsubseteq$ ≤1.(is_next(G,G2) $\sqcap$ end_state(G,SUCCEED)) $\sqcup$ ≤1.(is_next(G,G2) $\sqcap$ end_state(G,FAIL)) $\sqcup$ ≤1.(is_next(G,G2) $\sqcap$ end_state(G,VIOLATE))	A Goal can have no more than one immediate successor in the event of a specific ending state
G6	Goal $\sqsubseteq$ ∃follows.Goal	A Goal can only be followed by another Goal
G7	Goal(G) $\sqsubseteq$ ¬follows(G,G)	A Goal cannot follow itself (no loops)
G8	isNext $\sqsubseteq$ follows	A Goal follows another Goal if it is the next Goal
G9	follows ∘ follows $\sqsubseteq$ follows	follows is transitive (if follows(A,B) and follows(B,C), then follows(A,C))
G10	includes ≡ startsWith ∘ follows	All Goals in a Mission must *potentially* follow the starting Goal (satisfiability vice entailment)

lationships are fully and formally represented. The mission is expressed in RDF/OWL syntax and logically validatable. Thus, semantic validation of human orders is possible using Semantic Web systems, providing a significant capability for building human trust in safe operations by unmanned systems.

The ability to provide a full description of all goals and constraints within the MEO using vehicle-executable code significantly strengthens the already-powerful MEA construct. Specifically, not only is it impossible to define a mission for a particular vehicle without explicit `canExecute` relationships between the vehicle and all mission goals and `canIdentify` relationships between the vehicle and all mission constraints, but it is also impossible to assert these relationships without an appropriate vehicle-specific encoding of all mission goals and constraints. Once again, human intent to meet all ethical constraints without self-contradicting guidance is formally confirmed through semantic validation.

Finally, automated reasoning with DL-based ontologies is an important tool for ensuring strategic level mission validity before conducting exhaustive verification, validation, and accreditation (VV&A) testing using virtual simulators and real-world operations. If, for instance, an attempt is made to finalize a mission that includes goals that are not executable by the target vehicle, an OWL/RDF reasoner can quickly identify this shortcoming using the MEO. Similarly, a reasoner can detect mission flow-graph structural errors based on ontology rules that preclude illogical loops, unreachable goals, or untasked/orphan goals without specified predecessors or successors. A reasoner can also simplify the mission definition process by detecting implicit relationships that are not explicitly or correctly specified. For example, if a particular `Goal` entity is defined and is reachable from the mission's start goal (i.e., the one with which the containing `Mission` has a `startsWith` relationship) through a sequence of goal successes and failures, then it must be in an `includes` relationship with that particular mission whether the relationship is explicitly declared or not.

As described, a DL-based MEO defined in OWL ties the MEA semantics discussed in previous sections to actual target vehicles. The ontology ensures not only the validity of the mission structure for arbitrary strategic level mission flow graphs, but their executability on the particular target vehicles as well. Thus, all of the requirements originally posed for assignment of human responsibility—that the mission is defined in a mathematically rigorous and fully-understood manner, that the mission specification is equally understandable by the human operator and the target vehicle, and that the mission is comprised entirely of trusted vehicle behaviors—are fully specified in the human-approved mission orders and enforced by the strict semantics encoded in the ontology. The relations are defined in such a way that it permits simple customization for specific robot types.

D. Formal Specification of the Ontology

The principal concepts and relationships represented in the mission ontology are shown in Fig. 12. Definitions of the full set of concepts and roles (also known as classes and properties) in this ontology satisfy a number of rules, as specified in Table II, to support logical inferences relating to the validity of the mission structure.

E. MEO Summary

It is possible to produce general robot mission orders that are understandable by (legally culpable) humans and are reliably and safely executable by robots. The semantic representation of the mission plans permits automated examination of the plans for logical consistency and provides an enhanced methodology for software implementations to process missions. Even if perfectly executable, proper robot logic is not useful in military context unless it is a directly compatible extension of warfighter logic. Rules of engagement (ROEs), concepts of operation, doctrine, tactics, etc., must be expressible in equivalent terms to be effective and usable. Constraint tests must be determinable by a human supervisor or critic, by a virtual environment running a simulation, or by on-board robot sensors in the operating environment. Constraint tests can match common guidelines such as rules of the road, water-space management, ROEs, operational orders, and other expressions of bounds on mission conduct. These expressions cannot be vague, must result in clear logical determinism (true or false), must be able to combine multiple logical constraints, and need to note reporting requirements when human permission is necessary. For strategic-level task controllers, the ternary tactical task sequencer using ethics constraints may allow traceability and accountability for the full set of executed robot tasks without loss of generality. ROEs and other expressions provide ethical constraints and boundary conditions on robot strategic planning and operational conduct that can work cooperatively and satisfactorily with humans.

VI. Conclusion

Humans given authority over potentially lethal robotic systems must be provided with realistic capabilities that enable meaningful supervision, responsibility, and accountability. AVCL mission definitions provide one such example: the ability to define strategic level goals and tactical level tasks in a human-understandable way, and in syntactically validatable form, that a wide variety of robots might interpret and execute. MEA formalisms show that the underlying programming constructs are tractable and sufficiently general to ensure broad feasibility. MEO validation provides further abilities to logically evaluate the semantic correctness and completeness of ethically constrained mission definitions. Together these capabilities provide a practical framework for ethically grounded human supervision of unmanned systems.

Specific conclusions and recommendations include the following.

1) Ethical operation of autonomous systems requires human responsibility, accountability, and understanding. Any decision to deploy potentially lethal force without appropriate constraints or control may be dangerous, immoral, and illegal.
2) Lethality requires an ethical and legal basis for unmanned system operations. Principles such as vicarious liability clearly show that humans are both responsible and also vulnerable. Robot self-preservation becomes irrelevant when human life is at stake.

3) Unmanned systems can remain supervised and semiautonomous, even if communications are lost, if appropriate guidance and checkpoints are provided.
4) Human clarity and cooperative action are essential for supervising robots together with human teams.
5) Unmanned systems can be compatibly tasked in concert with human teams.
6) Applied ethics equals defining tasks and observing constraints before executing potentially harmful tasks.
7) The mathematical concepts of DLs and ontologies, as implemented by Semantic Web technologies, is proposed to capture common logical and ethical relationships for mission and task definition.
8) A mission-definition approach to constrained tasking is actionable for all unmanned systems regardless of software architecture.
9) For those choosing to adopt RBM for robot control, we urge that the strategic level be fail-safe and exhaustively testable.
10) Ethical constraints on robot mission execution are possible today. There is no need to wait for notional future developments in AI. It is therefore a moral imperative that ethical constraints in some form be introduced into the software of all robots capable of inflicting unintended harm to humans or property.

Ethical operation of robotic systems requires human accountability. In both the legal and moral sense, this implies that human operators be in a position to understand, and therefore control, robot mission outcomes. This level of understanding can be achieved through the satisfaction of three requirements: operator understanding of high-level mission flow, mission descriptions understandable to both human operators and target vehicles, and mission descriptions consisting entirely of trusted behaviors and constraints.

Algorithms cannot replace human responsibility. Even so, a fully testable technology (such as that provided by the MEA and MEO formalisms) allows for the assignment of human accountability. Specifically, the MEA provides a mathematically rigorous mechanism for mission definition and execution as an exhaustively testable flow diagram. This approach ensures that accountable operators can fully understand all high-level task sequences before authorizing robot operations. The MEO employs DLs and Semantic Web technologies to provide strong assurances that MEA mission definitions are semantically correct and fully executable by specific target vehicles.

By applying the best strengths of human ethical responsibility, repeatable formal logic and directable unmanned systems together, these capabilities provide a practical framework for ethically grounded human supervision of unmanned systems. Much important work awaits.

Acknowledgment

The authors gratefully acknowledge the fundamentally important insights and continuing contributions of many dozens of NPS students and colleagues. Multiple external reviews helped improve this work and are much appreciated. The content of this paper reflects the opinions of the authors and not necessarily the position of the NPS or the Department of the Navy.

References

[1] D. T. Davis, *Design, Implementation, and Testing of a Common Data Model Supporting Autonomous Vehicle Compatibility and Interoperability*. Monterey, CA, USA: Naval Postgraduate School, 2006.

[2] D. P. Brutzman, D. T. Davis, G. R. Lucas, Jr., and R. B. McGhee, "Run-time ethics checking for autonomous unmanned vehicles: Developing a practical approach," in *Proc. 18th Int. Symp. Unmanned Untethered Submersible Technol.*, Portsmouth, NH, USA, 2013, pp. 78–89.

[3] D. Brutzman *et al.*, "Autonomous vehicle command language," 1 Jan. 2013. [Online]. Available: https://savage.nps.edu/Savage/AuvWorkbench/AVCL/AVCL.html. Accessed on: Dec. 9, 2016.

[4] D. Brutzman, "A virtual world for an autonomous underwater vehicle," Naval Postgraduate School, Monterey, CA, USA, 1994.

[5] D. P. Brutzman, D. T. Davis, C. L. Blais, and R. B. McGhee, "Ethical mission definition and execution for maritime and naval robotic vehicles: A practical approach," in *Proc. IEEE/MTS Oceans*, Monterey, CA, USA, 2016, doi: 10.1109/OCEANS.2016.7761348.

[6] K. Čapek, *Rossum's Universal Robots*, 2004 ed., New York, NY, USA: Penguin, 1921.

[7] I. Azimov, *I, Robot*, New York, NY: Bantam Dell, 1950.

[8] R. Sparrow, "'Just say No' to Drones," *Technol. Soc. Mag.*, vol. 31, no. 1, pp. 56–63, 2012.

[9] R. Arkin, *Governing Lethal Behavior in Autonomous Robots*. Boca Raton, FL, USA: Taylor & Francis, 2009.

[10] P. Lin, K. Abney, and G. A. Bekey, Eds., *Robot Ethics: The Ethical and Social Implications of Robotics*. Cambridge, MA, USA: MIT Press, 2011.

[11] P. Scharre, "Autonomous weapons and operational risk," Feb. 2016. [Online]. Available: http://www.cnas.org/autonomous-weapons-and-operational-risk

[12] Merriam-Webster, "Vicarious liability," in *Merriam-Webster's Dictionary of Law*, Springfield, MA, USA: Merriam-Webster, 2011.

[13] Department of Defense, "Autonomy in weapon systems, directive 3000.09," Nov. 21, 2012. [Online]. Available: http://www.dtic.mil/whs/directives/corres/pdf/300009p.pdf. Accessed on: Dec. 23, 2014.

[14] W. P. Mack, H. A. Seymour, and L. A. McComas, *The Naval Officer's Guide*. 11th ed., Annapolis, MD, USA: Naval Inst. Press, 1998.

[15] A. L. Schuller, "At the crossroads of control: The intersection of artificial intelligence in autonomous weapon systems with international humanitarian law," *Harvard Nat. Sec. J.*, vol. 8, no. 2, pp. 379–425, May 30, 2017.

[16] A. L. Schuller, "Inimical inceptions of imminence: A new approach to anticipatory self-defense under the law of armed conflict," *UCLA J. Int. Law Foreign Affairs*, vol. 18, no. 2, pp. 161–206, 2014.

[17] D. P. Brutzman, "Network optional warfare (NOW)," Naval Postgraduate School, Jan. 6, 2014. [Online]. Available: https://wiki.nps.edu/display/NOW/Network+Optional+Warfare. Accessed on: Dec. 1, 2017.

[18] D. Brutzman, T. Healey, D. Marco, and B. McGhee, "The phoenix autonomous underwater vehicle," in *AI-Based Mobile Robots*, Cambridge, MA, USA: MIT/AAAI Press, 1998.

[19] D. Marco, A. Healey, and R. McGhee, "Autonomous underwater vehicles: Hybrid control of mission and motion," *Autonom. Robots*, vol. 3, pp. 169–186, 1996.

[20] R. Byrnes, *The Rational Behavior Model: A Multi-Paradigm, Tri-Level Software Architecture for the Control of Autonomous Vehicles*. Monterey, CA, USA: Naval Postgraduate School, 1993.

[21] R. B. Byrnes, A. J. Healey, R. B. McGhee, M. L. Nelson, S. Kwak, and D. P. Brutzman, "The rational behavior software architecture for intelligent ships," *Naval Eng. J.*, vol. 108, pp. 43–56, Mar. 1996.

[22] C. N. Duarte *et al.*, "A common control language to support multiple cooperating UAVs," in *Proc. 14th Int. Symp. Unmanned Untethered Submersible Technol.*, Durham, NH, USA, 2005.

[23] M. Ricard and S. Kolitz, "The ADEPT framework for intelligent autonomy," in *Proc. Intell. Syst. Aeronautics Workshop*, Brussels, Belgium, 2002, pp. 1-1–1-11.

[24] J. Albus, "Engineering intelligent systems," in *Proc. IEEE ISIC/CIRA/ISAS Joint Conf.*, Gaithersburg, MD, USA, 1998.

[25] D. P. Brutzman, R. B. McGhee, and D. T. Davis, “An implemented universal mission controller with run time ethics checking for autonomous unmanned vehicles–A UUV example,” in *Proc. OES-IEEE Autonom. Underwater Veh.*, Southampton, U.K., 2012. doi: 10.1109/AUV.2012.6380744.

[26] C. Dannegger, “Real-time autonomic automation,” in *Springer Handbook of Automation*, S. Y. Nof, Ed., New York, NY, USA: Springer-Verlag, 2009, pp. 381–404.

[27] G. T. Hammond, *The Mind of War: John Boyd and American Security*. Washington, DC, USA: Smithsonian Inst. Press, 2001.

[28] N. Rowe, *Artificial Intelligence Through Prolog*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1988.

[29] M. Minsky, *Computation: Finite and Infinite Machines, Englewood Cliffs*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1967.

[30] C. Petzold, *The Annotated Turing: A Guided Tour through Alan Turing’s Historic Paper on Computability and the Turing Machine, Indianapolis*, Indianapolis, IN, USA: Wiley, 2008.

[31] S. G. Simpson, “Logic and mathematics,” in *The Examined Life: Readings from Western Philosophy from Plato to Kant*, S. Rosen, Ed., New York, NY, USA: Random House, 2000.

[32] R. B. McGhee, D. P. Brutzman, and D. T. Davis, “A universal multiphase Mission Execution Automaton (MEA) with prolog implementation for unmanned untethered vehicles,” in *Proc. 17th Int. Symp. Unmanned Untethered Submersible Technol.*, Portsmouth, NH, USA, 2011, pp. 241–250.

[33] D. Brutzman *et al.*, “Autonomous unmanned vehicle (AUV) workbench,” Sep. 18, 2016. [Online]. Available: https://savage.nps.edu/AuvWorkbench. Accessed on: Dec. 9, 2016.

[34] R. B. McGhee, D. P. Brutzman, and D. T. Davis, “Recursive goal refinement and iterative task abstraction for top-level control of autonomous mobile robots by mission execution automata–a UUV example,” Naval Postgraduate School, Monterey, CA, USA, 2012.

[35] R. B. McGhee, D. P. Brutzman, and D. T. Davis, “A taxonomy of turing machines and mission execution automata with lisp/prolog implementation,” Naval Postgraduate School, Monterey, CA, USA, 2011.

[36] S. S. Skiena, *The Algorithm Design Manual*, 2 ed., London, U.K.: Springer-Verlag, 1997, pp. 169–178.

[37] U.S. Army Training Doctrine Command Analysis Center, “COMBATXXI,” Sep. 29, 2015. [Online]. Available: http://www.trac.army.mil/COMBATXXI.pdf. Accessed on: Jan. 28, 2016.

[38] S. Posadas, “Stochastic simulation of a commander’s decision cycle (SSIM CODE),” M.S. thesis, Operations Research Naval Postgraduate School, Monterey, CA, USA, 2001.

[39] T. Scholz, “The state transition diagram with path priority and its applications,” Naval Postgraduate School, Monterey, CA, USA, Sep. 1993.

[40] M. Ortiz and M. Šimkus, “Reasoning and query answering in description logics,” in *Reasoning Web. Semantic Technologies for Advanced Query Answering (Lecture Notes in Computer Science)*, vol. 7487, Berlin, Germany: Springer-Verlag, 2012.

[41] T. Berners-Lee, J. Hendler, and O. Lassila, “The semantic web,” *Sci. Amer.*, vol. 284, no. 5, pp. 34–43, May 2001.

[42] World Wide Web Consortium, “Web Ontology Language (OWL) Semantic Web Standards,” Dec. 11, 2013. [Online]. Available: https://www.w3.org/2001/sw/wiki/OWL. Accessed on: Dec. 9, 2016.

[43] World Wide Web Consortium, “Resource Description Framework (RDF) semantic web standards,” Mar. 15, 2014. [Online]. Available: https://www.w3.org/2001/sw/wiki/RDF. Accessed on: Dec. 9, 2016.

[44] I. Horrocks, “Ontologies and the semantic web,” *Commun. ACM*, vol. 51, no. 12, pp. 58–67, 2008.

[45] M. D. Daconta, L. J. Orbst, and K. T. Smith, *The Semantic Web, A Guide to the Future of XML, Web Services, and Knowledge Management*. Indianapolis, IN, USA: Wiley, 2003.

[46] D. T. Davis, “Semantic web and inferencing technologies for department of defense systems,” Naval Postgraduate School, Monterey, CA, USA, 2014.

[47] “Uniform resource identifier (URI),” Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Uniform_Resource_Identifier. Accessed on: Dec. 1, 2017.

[48] A. Krisnadhi, F. Maier, and P. Hitzler, “OWL and Rules,” in *Reasoning Web: Semantic Technologies for the Web of Data*, A. Polleres, M. d’Amato, S. Arenas, S. Handschuh, P. Kroner, S. Ossowski, and P. F. Patel-Schneider, Eds., Heidelberg, Germany: Springer-Verlag, 2011, pp. 382–415.

[49] M. Horridge, “A practical guide to building OWL ontologies using protege 4 and CO-ODE tools. Edition 1.3.,” Mar. 24, 2011. [Online]. Available: http://owl.cs.manchester.ac.uk/publications/talks-and-tutorials/protg-owl-tutorial. Accessed on: Dec. 23, 2014.

[50] G. Lokhorst and J. ven den Hoven, “Responsibility for military robots,” in *Robot Ethics: The Ethical and Social Implications of Robotics*, N. G. Lin, K. Agney, and G. A. Bekey, Eds., Cambridge, MA, USA: MIT Press, 2012, pp. 145–156.

[51] C. Langley and C. Spenser, “The visual development of rule-based systems,” *PC AI Mag.*, vol. 18, no. 3, pp. 29–36, 2005.

[52] Stanford University, “Protege ontology editor and application framework,” Stanford Univ., Stanford, CA, USA, 2016. [Online]. Available: http://protege.stanford.edu. Accessed on: Dec. 9, 2016.

Don Brutzman (S’91–M’95) received the Ph.D. degree in computer science from Naval Postgraduate School, Monterey, CA, USA, in 1994.

He is currently a Computer Scientist and an Associate Professor of Applied Science in the Undersea Warfare Academic Group and the Department of Information Sciences, Naval Postgraduate School, Monterey, CA, USA. He is also the Co-Chair of the Extensible 3D (X3D) Graphics Working Group, Mountain View, CA, USA, and a Founding Member of the Board of Directors for the nonprofit Web3D Consortium, Mountain View, CA, USA. He is the lead author of the book *X3D Graphics for Web Authors* (San Mateo, CA, USA: Morgan Kaufmann, April 2007). He is a retired Naval Submarine Officer and the Principal Investigator for the Network-Optional Warfare and robodata projects. His research interests include underwater robotics, real-time three-dimensional computer graphics, artificial intelligence, and high-performance networking.

Curtis L. Blais (M’93) received the B.S. and M.S. degrees from the University of Notre Dame, Notre Dame, IN, USA, in 1972 and 1973, respectively, both in mathematics.

Since 1999, he has been a member of the research faculty at the Modeling, Virtual Environments, and Simulation Institute, Naval Postgraduate School (NPS), Monterey, CA, USA. He conducts research and teaches in modeling and simulation (M&S), encompassing simulation development processes, standards, human behavior modeling, and information modeling. Before NPS, he worked for 25 years in M&S, first as a Mathematician/Operations Research Analyst at the Space and Naval Warfare Systems Command, San Diego, CA, USA, and then as a Software Engineer, the Project Manager, and the Department Manager with the Systems Exploration Incorporated and Visicom. Over those years, he worked on numerous simulation projects, including development and delivery of several generations of command staff training systems for the U.S. Marine Corps, from 1976 to 1999. His current research interests include improvement of human behavior models to distinguish human performance from robotic system performance in analytical combat models.

Duane T. Davis received the Ph.D. degree in computer science from Naval Postgraduate School, Monterey, CA, USA, in 2006.

He is currently a Research Professor at the Department of Computer Science, Naval Postgraduate School, Monterey, CA, USA, and has been a member of the NPS faculty since 2008. His teaching and research interests include multivehicle robotic systems, swarm robotics, cybersecurity and operations, and ethical employment of autonomous and cyber capabilities. In addition to his instructional and research roles, he is the Academic Associate at the Cyber Academic Group and is responsible for the development and maintenance of learning objectives, educational goals, and course content for five cyber degree and certificate programs. Prior to his current position, he was a Career Naval Officer, for more than 20 years in the aviation community. While on active duty he held leadership positions in both operational and educational environments, and participated in deployments in support of numerous real-world operations.

Robert B. McGhee (F'90) was born in Detroit, MI, USA, in 1929. He received the B.S. degree in engineering physics from the University of Michigan, Ann Arbor, MI, USA, in 1952 and the M.S. and Ph.D. degrees in electrical engineering from the University of Southern California (USC), Los Angeles, CA, USA, in 1957 and 1963, respectively.

After graduation with his B.S. degree in 1952, he was with the U.S. Army Ordnance Corps as a Guided Missile maintenance Officer. Subsequently, beginning in 1955, he worked as a Guided Missile Engineer at Hughes Aircraft Company, Culver City, CA, USA, while also working toward the M.S. and Ph.D. degrees at USC. In 1963, he joined the Faculty of Electrical Engineering at USC, first as an Assistant Professor and subsequently as an Associate Professor. In 1968, he became a Professor of Electrical Engineering at Ohio State University, Columbus, OH, USA, where he remained for 17 years. In 1985, he joined the Naval Postgraduate School as a Professor of Computer Science until 2004, when he retired and was appointed Professor Emeritus. From the beginning of his career, his research activities have centered on various aspects computer engineering and robotics, where he has remained active up to the present time. For the past six years, he has been primarily concerned with developing mathematical concepts and practical software architectures capable of providing military robots with a capacity to observe ethical constraints on their behavior during mission execution, and able to refuse to execute illegal orders.