2018-04-30

Acquisition Challenges of Autonomous Systems

Tate, David M.; Sparrow, David A.

Monterey, California. Naval Postgraduate School

http://hdl.handle.net/10945/58700

SYM-AM-18-042

PROCEEDINGS
OF THE
FIFTEENTH ANNUAL ACQUISITION RESEARCH SYMPOSIUM

WEDNESDAY SESSIONS
VOLUME I

Acquisition Research:
Creating Synergy for Informed Change

May 9–10, 2018

Published April 30, 2018

Prepared for the Naval Postgraduate School, Monterey, CA 93943.

ACQUISITION RESEARCH PROGRAM
GRADUATE SCHOOL OF BUSINESS & PUBLIC POLICY
NAVAL POSTGRADUATE SCHOOL

Acquisition Challenges of Autonomous Systems

David M. Tate—joined the research staff of the Institute for Defense Analyses' (IDA) Cost Analysis and Research Division in 2000. Prior to that, he was an Assistant Professor of Industrial Engineering at the University of Pittsburgh, and the Senior Operations Research Analyst—Telecom for Decision-Science Applications, Inc. At IDA, he has worked on a wide variety of resource analysis and quantitative modeling projects related to national security. These include an independent cost estimate of Future Combat Systems development costs, investigation of apparent inequities in Veterans' Disability Benefit adjudications, and modeling and optimization of resource-constrained acquisition portfolios. Tate holds bachelor's degrees in philosophy and mathematical sciences from the Johns Hopkins University, and MS and PhD degrees in operations research from Cornell University. [dtate@ida.org]

David A. Sparrow—received a PhD in physics in 1974 and spent 12 years as an academic physicist. He joined IDA in 1986 and has been a Research Staff Member ever since, with brief forays into management and government service. He was the first Director of the IDA Simulation Center from 1989 to 1990, and Assistant Director of the Science and Technology Division from 1993 to 1997. He then joined the government for a two-year stint as Science Advisor on Modeling and Simulation to the Director, Operational Test and Evaluation. Since returning to IDA, he has focused on technical issues in system development, especially ground combat systems—expansively defined to include unexploded ordinance (UXO), counter mine, and, occasionally, missile defense. He has authored ~100 refereed papers and invited talks on various academic and national security topics. [dsparrow@ida.org]

Abstract

The Department of Defense has stated publicly that future defense capabilities will depend strongly on autonomous systems—systems that make sophisticated judgments about the world and choose appropriate courses of action, and perhaps even adapt and learn over time. Developing and deploying such systems poses more than just a technical challenge in robotics and artificial intelligence—it also poses many challenges to the acquisition process and workforce. From cost estimation to sustainment planning, every aspect of acquisition will be affected. Test and evaluation, in particular, may require not only novel methodologies and resources, but organizational and process changes as well.

Acquiring Autonomy—Bottom Line Up Front

We consider the life cycle of a typical major acquisition program, and identify the processes and activities that are complicated by the presence of autonomy. We argue that every aspect of acquisition planning, management, and execution will be more difficult and less certain for systems with autonomous capabilities—and significantly more so for some of the ambitious autonomous capabilities currently envisioned by the Department of Defense (DoD) and emphasized in the National Defense Strategy (DoD, 2018). Designing and implementing the autonomous capabilities will force a different approach to system development and program management than is customary—simultaneously requiring more rigor and more flexibility.

Because there will be a lack of historical precedent to guide planning, execution, and oversight activities, a number of key “control loops” within the development effort will be new, different, and/or more difficult:

- Diagnosis of performance issues will be harder and will thus take longer than for non-autonomous systems.
- Determining and implementing corrective design changes will be harder and may require simultaneous changes to hardware, software, and concepts of operations.
- The division of responsibilities between humans and machines, and the protocols and concepts of operations (CONOPS) that enable effective teaming, will necessarily be part of the system design, rather than something to be figured out and perfected after the system has already been designed and built.
- Achieving acceptable performance will almost certainly involve *iterative experimentation* of a kind usually found only in Science and Technology (S&T) or Advanced Concept Technology Demonstration (ACTD) projects.
- Developmental Test and Evaluation (DT&E) and Operational Test and Evaluation (OT&E) will both require more frequent tests and new kinds of test instrumentation. Testing will also need to be more closely coupled with contractor and Program Office design processes than is typical.
- Achieving acceptable performance will require representative human users/operators/teammates far earlier and more frequently in the development cycle than is common under current practice.
- Developmental and regression testing will continue throughout the acquisition life cycle, often including occasional post-fielding regression testing.

The features of autonomous capability that will drive these changes include the following:

- The complexity and general lack of transparency of the core artificial intelligence (AI) modules enabling autonomy: *perception, reasoning, course of action, selection,* and *adaptation*;
- Substantive Human-Machine Teaming, involving shared situational awareness and understanding of mission objectives between human and machine agents; and
- The potential for undesired emergent behaviors of the complex system.

The literature on autonomous capabilities tends to focus on technical challenges of how to implement autonomy. Very little literature exists on the practical aspects of turning promising new AI technologies into commercial products or effective and suitable government systems that authorities will be willing to see fielded. We consider the technical aspects of implementing autonomy only to the extent that they can be expected to affect acquisition success—that is, timely and affordable delivery of effective and suitable systems. The main body of the paper discusses how the interactions among these AI modules and between the AI modules and the human team members, coupled with certain anachronistic features of the DoD acquisition system, create challenges throughout the acquisition process.

Why Autonomy Breaks Acquisition

How Autonomy Works

What do we mean by “autonomous systems”? One way to think about this is in terms of the “OODA loop” first described by Col. John Boyd (1995). Human beings performing complex tasks in complex environments repeatedly:

- **Observe**—take in data about the environment and themselves
- **Orient**—use that information to create a mental model of what is going on
- **Decide**—identify possible courses of action and choose one
- **Act**—implement the decision

Autonomous systems are those that implement a nontrivial OODA loop of their own, especially in the Orient and Decide modes. They collect sensor data for their own use, they process the data they collect to maintain a complex world model describing their environment and current state, and they develop possible courses of action and select which action to implement without direct human instruction.

To distinguish the machine version of OODA from the human version (and to emphasize the ways in which it is different), we will use a slightly different terminology. The corresponding capabilities that enable autonomy are

- **Perception**—collecting data about the environment and making sense of it; building a world model
- **Reasoning**—extrapolating from the world model to interpret events, intentions, unobserved entities, etc.
- **Selection**—identifying available courses of action and choosing one

Some of these capabilities are more sophisticated versions of familiar system features. Perception, for example, can be thought of as a natural extension of computer vision and sensor fusion capabilities. Selection allows more nuanced and complex behaviors than past state-action lookup tables.

These defining capabilities are in turn enabled by a wide array of specific AI methods and algorithms. These include various forms of Machine Learning (ML)—supervised learning, unsupervised learning, reinforcement learning—combined with complex constraint processing, theorem-proving, and other Reasoning techniques. The ML subsystems are implemented using specific learning architectures, such as Deep Learning or Generative Adversarial Networks (GAN), and their training is accomplished using specialized optimization techniques such as backpropagation.

The result, particularly for neural network-based forms of ML, is a system that does not so much process information algorithmically, but instead reaches a snap judgment when presented with an input. The system has “hunches”; the purpose of the training is to make those hunches accurate. In general, these hunch-making systems will be nested and combined, with feedback loops that make it essentially impossible to trace the “logic” of how the output relates to the input. This lack of transparency turns out to have important consequences for acquisition.

Engineering Design vs. Experimentation and Discovery

If you want to build a bridge, you start by deciding how much traffic of what type the bridge will be required to carry. You consider the geology of the planned site, the force of the current of the river you are bridging, the prevailing winds, and many other factors. You then consider possible bridge designs that could carry that much load under those conditions, and decide which design best meets your needs, taking into consideration cost, useful lifetime, maintenance required, time to build, and other measures. The key here is that you have a very good idea before you build the bridge just how much load it will be able to carry. You don't have to experiment; there is no trial-and-error involved. The only testing you need to do is to confirm that you built the bridge you designed—proper materials and processes, correct measurements, and so forth.

If you want to build an autonomous system to perform a given set of missions, there is (at present) no corresponding engineering science you can rely on. For any but the simplest missions, you can't look at the performance requirements and know that if you use Algorithm A, trained on training data set B, with decision logic C, the system will perform at overall level X. If the autonomous system is going to interact significantly with humans in a relationship that could be characterized as "teaming," even very simple missions can require considerable experimentation and fine-tuning before achieving the desired overall level of mission performance.

Consider the case of early aircraft autopilot systems. It is fairly straightforward to design a system that can maintain a given bearing and altitude without human intervention. It proved to be much harder to design a CONOPS and protocol for human-autonomy interaction that allowed humans and autopilots to cooperate smoothly without occasionally crashing an airplane.

The value of engineering design is that it completely characterizes the behavior of a proposed system. This means not only that you can know what a system can do without actually having to build it, but that you can know what that system will *not* do. In the absence of a well-established body of knowledge that supports predictive engineering design, all system development will need to rely on a process of experimentation and discovery—not only to figure out a design that works, but also to establish the dependability of that design.

For the program manager (PM), the presence of significant experimentation during development eliminates an important breakpoint that is built into the acquisition system. There are critical differences between activities before and after Milestone (MS) B—the Technology Maturation and Risk Reduction phase prior to MS B is about technology development, whereas the Engineering and Manufacturing Development (EMD) phase after MS B is about product and manufacturing process development. Technology development is inherently event-driven, but the theory has been that by requiring mature technologies at MS B, it is possible to make EMD predictable in cost and schedule. The need for experimentation as part of the design of autonomous systems pushes the inherently event-driven elements of the program well into EMD, where the supporting activities of Systems Engineering (SE), DT&E, and Project Management are ill-equipped to deal with it.

Proving a Negative Is Hard

For most current systems, safety and cybersecurity requirements are the only requirements of the form "the system must NOT ___." These proscriptions are typically not treated as "requirements" (in the Capability Development Document Key Performance Parameter sense) at all, but rather as "certifications" to be granted by stovepiped authorities separate from both the developers and the Test and Evaluation (T&E) processes. In addition, the scope of what is considered a "safety issue" is quite limited. A rifle program's safety certification will be focused on ensuring that the rifle will not explode in the operator's face, will not cause burns during normal operations, and will fire bullets only in the direction it is being aimed. The certification will *not* be concerned with the possibility that the rifle could be fired at friendly forces, stolen and used against noncombatants, or deliberately wedged into the hinges of a troop transport—those are risks to be mitigated by personnel screening and training, not by design of the rifle.

For autonomous systems, many unwanted outcomes that are typically avoided through proper screening and training of human operators will have to instead be avoided through the design of the system. Compliance with these "negative requirements" is inherently more difficult to demonstrate, for the familiar reason that you cannot prove a negative. It is easy to demonstrate that a skilled marksman can hit a six-inch target at 400 meters, by having a marksman do that. Reliability is more difficult, but can be demonstrated statistically. However, demonstrating that something will *never* happen is much more challenging. We don't even attempt to demonstrate that an infantryman will never shoot a civilian—accidentally or deliberately—but an autonomous system may well require us to provide convincing evidence of that.

In practice, we do not require proof that unacceptable outcomes will not occur; we require reasonable confidence. This makes T&E of autonomous systems an exercise in risk management; however, this is very different from the typical 5-by-5 "risk cube" approach typically used in defense program management. The familiar approach focuses on *identified* risks that are both serious and reasonably likely (~5% probability) to occur. In the case of complex autonomous capabilities, the program will need to produce evidence that the *unidentified* risks that may be catastrophic are extremely rare (multiple zeros in the probability). Further, in general, the argument supporting this assertion will need to be persuasive outside the program office.

State Space Explosion, Design of Experiments, and Autonomy

If you were given the task of testing a new passenger automobile tire, it would be a mistake to only test the tire on dry, straight, smooth, asphalt roads. Anyone who has done much driving knows that tires also need to be able to cope with curves, water (or ice), a variety of paving materials and surface conditions, and perhaps a range of temperatures. All of those things occur naturally in typical driving, and all of them matter.

For some systems, the set of parameters that matter, and the set of values they can take, is small enough that you could actually test every combination, to verify that the system performs acceptably in all of them. Similarly, for sufficiently simple software, it is possible to explicitly test every possible execution path of the software, to verify that the application behaves as intended. This is *exhaustive testing*. It is not very efficient, but it conveys a very high degree of confidence in the dependability of the system that was tested. The set of all possible combinations of relevant parameters is called the *state space* of the system in question.

For our automobile tire example, there is no chance that you could test every single possible combination of surface, surface condition, temperature, moisture, curviness, and so forth. This is *state space explosion*—when the number of configurations of interest grows faster than our testing needs can handle. However, we know the range of possible values each of those parameters can take, and we have a very strong physics-based understanding of which parameters are important and how performance varies with these factors and their interactions. It is possible to infer what a test of every possible combination would reveal by testing a much smaller number of well-chosen combinations of the parameters, using statistical inference to draw conclusions about how the tire would perform in situations *between* the ones that were explicitly tested. Design of Experiments (DOE) is the branch of statistics that studies how to do this efficiently and effectively, choosing a minimum set of design points to characterize system performance everywhere in the state space. DOE has been incredibly important in establishing the effectiveness and suitability of military systems with increasingly large state spaces.

DOE only solves the problem of state space explosion when

- We know which parameters are important.
- There aren't too many important parameters.
- We can reasonably expect that changes in system performance will be smooth in the regions of the state space that are between design points.
- We know in advance which regions of the state space are likely to exhibit rapid changes in performance for small changes in the parameters.

Unfortunately, those will generally not all be true for autonomous systems.

As noted previously, we have no engineering design theory for autonomy the way we do for tires, or for ships or aircraft or missiles. We don't know which inputs to the autonomy software will be important, we can't exhaustively test all of the possible execution paths of the software, and we can't statistically characterize performance over the entire state space using DOE. We are going to have to do something else—something that the Defense Acquisition System assumes you are never, ever going to do. We are going to have to experiment—a lot—at every stage of the acquisition process, probably including post-fielding sustainment.

The PM is then confronted with the following situation—the state space is too large for comprehensive exploration. It is not well enough understood for purely mathematical or statistical approaches such as DOE to ensure adequate coverage. The program office will need to use exploratory tools that are designed to preferentially find areas of potential concern, dynamically guiding the state space exploration in both simulation and real testing. There are a number of such approaches in development, but none that have become part of the standard development or T&E toolkit.

Emergent Behavior and Transparency

Emergent behavior is the general term for high-level behavior exhibited by a system that is hard to predict from the characteristics of the individual elements of the system. In physics, a classic example is trying to predict whether a given molecule will behave as a solid, liquid, or gas at a given temperature and pressure. Similarly, the tendency of snowflakes to exhibit hexagonal symmetry is an emergent behavior of water.

Human systems also exhibit emergent behavior. True gridlock—that state of urban traffic in which nobody can move—is an emergent behavior of individual driver behavior. In satellite communication systems, the tendency of speakers to speak and pause simultaneously is an emergent behavior of the combination of normal human speech patterns and the delay inherent in the communications system.

Autonomous systems, and autonomous systems interacting with humans, are even more prone to undesired emergent behavior than human systems are. Avoiding unwanted emergent behavior will be an important part of system development—which brings us back to the problem of proving a negative. Furthermore, when emergent behavior does arise, the problem of diagnosing the causes will be made much more complex by the inherent lack of transparency of the underlying algorithms in the autonomy.

For traditional systems, a common approach to avoiding unintended and undesired behaviors is to consider the worst-case scenarios of system behaviors, identify what states the system would have to be in for those behaviors to occur, then work backwards to determine how those states could be reached during actual operation of the system. For autonomous systems, there are severe limits on the effectiveness of this approach. In particular, the backwards causal trace of what conditions could lead to the undesired state may not be feasible if the undesired state is in part the output of a “black box” ML algorithm. Characterizing which inputs to, for example, a Deep Learning network would lead to specified outputs is not generally possible.

This lack of transparency is a big problem for verification and validation (V&V) of ML-based capabilities, even without the added issue of emergent behavior. For example, suppose that a target identification and cueing system is sometimes failing to warn its human teammates of a certain type of threat. Is the problem that the sensors are not seeing that threat? That the *perception* is not correctly identifying it? That the *reasoning* is concluding that it isn’t important? That the *selection* is incorrectly choosing not to report that target? Some combination of the above? If the *perception* is the problem, is the failure due to the algorithm being used, the input data used to train the ML, or bugs in the code?

We will need to find novel ways to instrument what (and how) the autonomy is thinking, if we are to be able to develop and certify these systems on useful timelines. This is a new requirement. We will also need to open up the black box of the software to DT&E. This is also new, and both technically and organizationally challenging. It is technically challenging because this is fundamentally a new type of instrumentation with few precedents. It is organizationally challenging in part because of intellectual property and trade secrets issues: The vendor may regard this “instrumentation” as an effort to acquire intellectual property that was not contracted for, or as putting intellectual property at risk of unauthorized disclosure.

Testing Autonomy Will Have to Be Different

Developmental Test and Evaluation (DT&E)

DT&E has many purposes, including

- To help produce the information necessary for efficient and successful development of the desired system capabilities
- To verify the adequacy of the system design
- To quantify reduction of technical risk
- To verify contract technical performance
- To certify readiness for OT&E

These purposes can be roughly lumped into three overarching categories:

- **Characterization**—how exactly is the system (or some subsystem) behaving?
- **Diagnosis**—why isn’t the (sub)system behaving properly?
- **Certification**—can we conclude that an interim performance goal for the system has been achieved?

What makes these categories distinct is that each of them requires *a different kind of testing*, collecting different measurements under different ranges of conditions. A common error in test planning is to assume that the same test event (or kind of test event) can support all of these disparate goals simultaneously. Just as a physician orders different kinds of blood tests, depending on whether the purpose is a general physical exam, diagnosis of a specific set of symptoms, or confirmation that a particular result has been achieved, so too DT&E must tailor its test designs to the particular purpose at hand. Early in the DT&E process, characterization will be primary. Late in the DT&E process, with luck, certification will be primary.

Autonomous capabilities complicate all three of these categories of DT&E. Autonomy is fundamentally a software-enabled capability, which means that autonomous systems inherit all of the T&E problems associated with complex software. In addition, the particular challenges posed by autonomous capabilities are not well addressed by traditional T&E practices, organizations, and resources. Although the kind of testing will vary with the characterization, diagnosis, or certification goals, there are new tools and new uses of existing approaches that will apply to all three. Although exhaustive testing is presumed infeasible, virtual testbeds allowing for extensive testing in a Live/Virtual/Constructive environment will be essential. The testing in this world will require the following:

- **Novel Modeling and Simulation (M&S) techniques**, enabling exploration of the decision space, rather than high fidelity representation of the physical space. Without this capability, characterization is impossible.
- **Novel instrumentation techniques**, enabling visibility into the perception, reasoning, and selecting functions. Without this capability, the data to support diagnosis (and ultimately certification) cannot be obtained.
- **New approaches to test design**, probably including adversarial test design, to allow efficient exploration of the extensive decision space.

The existing resources are not sufficient to support these activities, and in some cases include techniques that do not yet exist.

To be effective, DT&E will need to be more of a continuous engagement than a sequence of episodic events, with much more feedback into the architecture and design parts of development—the “D” in DT&E. This will require shorter test-redesign-test cycles that will probably continue much longer into the development cycle before morphing into the more familiar test-fix-test cycles. The “E” part of DT&E might require maintaining a detailed log of system and subsystem performance throughout the development cycle, to be used by downstream T&E and certification processes. This paradigm of continuous accumulation of evidence, rather than passing a convincing test event at the end of development, would be new—and is not supported by current organizational structures, division of responsibilities, or test resources.

When substantial human-machine teaming is intended, there will be additional challenges. The teaming CONOPS must be engineered into the machine. This will require active participation by users and user surrogates during the early stages of development. However, CONOPS development is not usually a program management office (PMO) responsibility, and the PMO may not have the authority to influence it.

With these challenges in mind, we turn back to the categories.

Characterization

For software systems, the question “What is the system doing?” is traditionally answered at the code execution level, by tracing the execution path and verifying that it is implementing the desired logic. If the complexity of the code is such that tracing the execution is impractical, or the logic of the algorithm is not understandable by humans, then some other way of interpreting what the system is doing will be required.

For systems with significant autonomous capabilities, much of the important system behavior will be implemented by algorithms that cannot be interpreted by humans as sequential logic. For example, any subsystem that relies on neural network models trained by supervised learning will not be executing sequential logic in the usual sense. Instead, when the neural network is presented with input data, it generates a “hunch” about the appropriate output, based on its training. Tracing the execution of the code that generates this output is unhelpful; it is merely a large number of weighted sums and transfer functions, uninterpretable at that level. Making sense of the hunch would require reverse-engineering the functioning of the neural network, in order to assign human-understandable meanings to patterns of weights in the network. This is a new requirement; we have not historically needed to instrument the internal states of mission software in order to decide whether it is performing well for the right reasons.

Diagnosis

A vital role of DT&E is to provide the measurements that enable testers to understand *why* the system is not behaving as intended. For software systems, this has traditionally meant finding bugs in the software that are causing it to implement incorrect logic.

For autonomous systems, because we have no engineering theory that would enable predicting system behavior from the software design, we cannot assume that undesired system behavior is being caused by bugs. The problem might just as easily be due to incorrect or inappropriate training data, a poor choice of algorithm, or unanticipated emergent behavior. If, for example, the system is failing to react appropriately to the actions of certain entities, it is difficult to tell from the outside whether the system is failing to see those entities (*perception*), failing to identify them correctly (*reasoning*), or failing to consider or choose the appropriate response (*selection*). If this is a system that continues to learn and self-modify after fielding, the system might have begun working correctly, but has learned a “bad habit” that is impairing performance (*adaptation*).

To diagnose the source of the problem, it will be necessary not only to instrument the internal states of the software, but also to have a normative model of what correct function looks like, in all modules and at multiple levels of descriptions, to compare the resulting measurements against. The instrumentation will be especially challenging on platforms that are highly constrained in available space, weight, or power—it will be difficult to *observe* system function without *distorting* system function. The development of normative models may be equally challenging, especially for enabling capabilities like ML, where proper function depends as much on how the model was trained as it does on correct implementation of the algorithm.

It is also important to mention that correct diagnosis does not always lead to a unique potential corrective action. If the *reasoning* functions are drawing incorrect conclusions from the world model built by the *perception* functions, it may not be obvious which module should be changed. Both may be functioning correctly according to the original design specification. Again, it will require experimentation to find the best way to achieve the intended functionality. Sometimes, the solution might involve adding a new runtime monitoring process that watches for problematic cases and intervenes when appropriate. This not only adds to the design complexity; it also introduces new modules that themselves will need to be verified and validated.

All of these challenges are exacerbated by an extreme version of the traditional chicken-and-egg problem posed by simultaneous hardware and software development. In general, software developers would prefer representative working hardware to test on, in order to verify that the software is working as intended. At the same time, hardware developers need representative working software to execute, to verify that the hardware is working as intended. Avoiding gridlock is not easy, especially if you can't be sure whether a given observed problem is due to hardware problems, software problems, or integration problems. Autonomous systems have all of these issues, but add the further complication of needing to develop the human-machine teaming CONOPS in parallel with both hardware and software. Correct diagnosis will require determining whether the hardware, the software, or the human team members are not behaving as intended, or whether the problem is in the algorithms chosen, the CONOPS design, or some combination of those things.

Certification

For autonomous or semi-autonomous weapon systems subject to DoD Directive (DoDD) 3000.09, *Autonomy in Weapon Systems* (DoD, 2017), there are certification requirements even before formal program initiation:

> Before a decision to enter into formal development, the USD(P), USD(AT&L), and CJCS shall ensure:
>
> *(1)* The system design incorporates the necessary capabilities to allow commanders and operators to exercise appropriate levels of human judgment in the use of force.

In other words, if DoDD 3000.09 applies, there are certifications about the design that must be made before development begins. The basis for these certifications is not yet well-defined.

In general, the T&E to support the various certification activities will involve open air testing of full up systems or major components. However, even these activities will require range safety releases, which are themselves a kind of certification. The *Joint Software Systems Safety Engineering Handbook* (DoD, 2010) addresses the relationship between autonomy embedded in the software and safety-critical functions.

For autonomous systems, safety release will generally depend upon a combination of a safety argument based on the entire development history and some specific "kill switch" function. A "kill switch" feature might itself depend on internal monitoring by an autonomous process, which could pose additional challenges, since internal monitoring may affect mission performance by competing with mission systems for power and computational resources.

Finally, the DT&E results are key inputs into Operational Test (OT) Readiness Reviews. While not a formal certification, readiness for OT typically requires the system under test to be "production representative." The current working definitions of "production representative" are all highly hardware-centric—they refer to tooling and production lines. For continually evolving complex software systems in general, the question of whether the system is yet production representative is difficult to answer. This is particularly true for ML systems. Many approaches to ML require extensive data for training. If the system were retrained using different data, it would exhibit different performance. Thus, the training data set itself needs to be production representative in some sense.

This vision for a persistent, intrusive instrumentation and experimentation process that includes CONOPS and training as design attributes is very different from what DT&E has looked like in the recent past. Even M&S, which is already familiar, would need to be used in novel ways for which little support currently exists. However, these are at least activities well within the traditional scope of a PMO's authority. Human teaming, human training, and CONOPS development are emphatically not part of a PM's traditional authority, and yet they will be essential elements of the system design. Safety issues introduced by the possibility of an incompetent or insane operator also demand novel T&E data collection, test designs, and information sharing across traditional organizational boundaries. This will introduce many of the same development issues as are seen when a program's design depends on decisions made in an external program—but in an unfamiliar context for which no administrative processes yet exist.

Operational Test and Evaluation (OT&E)

The purpose of operational testing is to determine whether or not a particular system is effective and suitable when used by warfighters in execution of their missions. The presumption is that DT&E has already established system performance over the intended range of operating conditions; all that needs to be confirmed is mission effectiveness and suitability when in the hands of intended users.

The scoping of OT under these conditions is straightforward—in some cases, a single scenario will suffice. For example, Air Superiority aircraft are usually tested in 1v1, 2v2, and 2v4 scenarios, but the number is manageable and predictable. For a ground vehicle, one would conduct both Major Combat Operations and Stability Operations scenarios, but again the scope is manageable and predictable.

For systems with autonomous capabilities, we have already seen that it may not be feasible to cover the state space during DT&E. For operational systems with substantial autonomous capabilities, there is no current understanding of how to scope the set of test missions by analogy to the air combat and ground combat examples. If suitability for autonomous systems requires the absence of unwanted emergent behavior, we do not know in general what would constitute sufficient evidence of that. In a previous paper (Tate et al., 2016), the authors argue that "sufficient evidence" might need to include the entire test history of the system, with the time series of improving performance and elimination of failure modes providing a degree of assurance not obtainable through pass/fail testing at the end of the development cycle.

Surprising uses of weapon systems by their operators have been observed during operational tests, even for systems with no autonomous capabilities. The probability of surprising emergent behavior is much higher when humans are teaming with autonomous systems rather than merely operating them, or when the systems are operating by themselves. This increases the probability of a test that cannot be conducted safely, or a test for which the wrong instrumentation or supporting data was available. As with DT&E, autonomous systems will generally require more test events in OT&E than traditional systems.

For systems subject to DoDD 3000.09, there are additional requirements that apply after IOT&E. In particular, Enclosure 2 of the Directive states,

> (b) After initial operational test and evaluation (IOT&E),any further changes to the system will undergo V&V and T&E in order to ensure that critical safety features have not been degraded.

> (1) A regression test of the software shall be applied to validate critical safety features have not been degraded. Automated regression testing tools will be used whenever feasible. The regression testing shall identify any new operating states and changes in the state transition matrix of the autonomous or semi-autonomous weapon system.
>
> (2) Each new or revised operating state shall undergo integrated T&E to characterize the system behavior in that new operating state. Changes to the state transition matrix may require whole system follow-on operational T&E, as directed by the Director of Operational Test and Evaluation (DOT&E).

In general, we do not yet know how to determine how much testing will be enough—or even whether the testing that has been done so far is sufficient. Even for systems for which DoDD 3000.09 does not apply, ongoing regression testing of software for safety and performance determination is certainly a best practice, where the same questions apply.

The earliest challenges for the PMO will be planning for the IOT&E. Scoping the tests will be more challenging than for systems without autonomous capabilities. In the case of robust teaming, scoping an OT that adequately explores the teaming arrangement remains an unsolved problem. The prospect of emergent behavior during the test may render the test unexecutable or uninformative. More time and resources will typically be required to execute the test events needed, and IOT&E will probably need to instrument more completely and archive more data in order to support post-fielding regression testing.

Cost and Schedule Estimation

Before you can do a meaningful Analysis of Alternatives (AoA), you need to have at least a rough guess at the cost, schedule, and operational effectiveness associated with each of the alternatives. (Ideally, you should also have a good idea of the risks and uncertainties associated with those attributes. The DoD does not have a stellar record in that regard.)

Standard cost and schedule estimating techniques were developed for sequential processes, in which the number, nature, and precedence relationships among tasks is known in advance. They work very well for routine construction projects. They work quite well for new system development when the new system is similar to past systems. They can even be useful for unprecedented new system development. They do not work well at all for projects with branching or looping logic, where the set of tasks and/or the number of times you will have to do each of them is not known with certainty.

In practice, cost estimators assume that experimentation is over—that the proposed design is (essentially) the design that will be built. They also tend to assume that the total cost of the program will be the sum of the costs of the components, failing to account adequately for critical path dependencies and costs of integration. The process of refining the design during development might involve test-fix-retest cycles to confirm that the design was implemented correctly, but it will not involve test-diagnose-redesign cycles at the higher levels of system architecture, design, or CONOPS.

This is already a bad assumption for software-intensive systems, first-of-a-kind science facilities, or complex system of systems integration (not to mention systems whose requirements keep changing over time). As noted above, autonomous systems partake of all three of those problematic categories, with the additional problem of necessary experimentation due to a lack of underlying engineering theory. Standard cost and schedule estimates for autonomous systems will always underestimate development time and resources required—and thus will underestimate cost as well. Until a substantial body of historical autonomy program data can be amassed, a new kind of estimating methodology will be needed, specially crafted to account for the uncertainty in how many diagnosis, redesign, and integration cycles will be required, to produce accurate forecasts of cost, schedule, and development risk.

To make matters worse, the cost and schedule estimates depend on the details of the CONOPS—how autonomy and human-autonomy teaming are to be used in performing the mission. The details of this CONOPS will not be known early in the program; they will necessarily be the result of substantial experimentation. This means that even this hypothetical novel estimating methodology, specially adapted for autonomous systems, will be subject to additional uncertainty due to reliance on educated guesses about key design features.

This suggests that it will be especially important for the PM to get updated cost and schedule estimates on an ongoing basis, keeping the cost estimators up to date on any changes in the autonomy or teaming design. In practice, although DoD guidance strongly encourages this kind of "living cost estimate," there are strong political and organizational incentives that drive programs to avoid doing that.

Conclusions

Current U.S. military strategy has placed great weight on developing and fielding advanced AI-enabled systems with autonomous capabilities that will allow these systems to operate themselves to a significant degree. They will also need to team with human warfighters as collaborating agents rather than as tools to be operated. We have shown that even if the technical challenges of implementing these new autonomous capabilities can be solved, they pose significant unprecedented challenges to the defense acquisition system, its organizations, and its processes.

At root, these challenges all arise from the absence of a mature scientific theory or engineering practice for autonomous systems that would enable designers to predict the macro-level behavior of an autonomous system from a description of its enabling algorithms and data. Until we can accurately predict how a given design will behave without building and testing it, we must instead develop systems through iterative experimentation and adjustment. At the same time, we must invent test, evaluation, and certification techniques that will enable reasonable assurance that a given autonomous system will perform dependably—safely, securely, effectively, reliably, etc.—within a specified range of mission contexts.

Establishing dependability with certainty would require proving a series of negatives—that the system will *not* behave unsafely, will *not* exhibit undesired emergent behavior, does *not* have exploitable cyber vulnerabilities, does *not* have exploitable training biases, does *not* have exploitable reasoning or selection algorithm foibles, and so forth. Since proving a negative is impossible, we recommend an approach inspired by Karl Popper's (2002) notion of falsification: namely, that a system can be considered dependable to the extent that we have tried as hard as possible to prove that it is *not* dependable and have failed. This approach would be a significant departure from current test practice, which is focused on doing the minimum possible amount of testing that can support a conclusion that a performance threshold has been met.

Implementing this approach will require novel T&E methodologies, such as intelligent adversarial testing in highly virtualized environments. These methodologies will in turn depend on T&E resources and infrastructure that do not yet exist—live/virtual/constructive simulation testbeds, instrumentation of AI internal states, sophisticated human/machine teaming support, copious data collection and archiving to establish dependability over time, etc. Such testing also implies developmental and operational test plans that are fundamentally unpredictable in duration and scope.

The defense acquisition system is predicated in large measure on the assumption that such unpredictability has been ironed out of a program by the time it passes Milestone B. That is generally not true even today; it will be far less true for the kind of experimental discovery processes needed to field effective autonomy. The DoD needs to be aware of that if they are serious about making autonomy a cornerstone of future military capability. Without significant changes to how we develop and test systems, the DoD will either not field autonomous systems, or will have very little reason to believe that its fielded systems will be safe, secure, and effective.

References

Boyd, J. R. (1995). *The essence of winning and losing* (Briefing).

DoD. (2010). *Joint software systems safety engineering handbook* [Ver. 1.0]. Retrieved from http://www.acqnotes.com/Attachments/Joint-SW-Systems-Safety-Engineering-Handbook.pdf

DoD. (2017). *Autonomy in weapon systems, incorporating Change 1* (DoD Directive 3000.09). Retrieved from http://www.dtic.mil/

DoD. (2018). *Summary of the 2018 national defense strategy of the United States of America: Sharpening the American military's competitive edge*. Retrieved from https://www.defense.gov/Portals/1/Documents/pubs/2018-National-Defense-Strategy-Summary.pdf

Popper, K. (2002). *The logic of scientific discovery*. New York, NY: Routledge. Retrieved from https://archive.org/details/PopperLogicScientificDiscovery (Original work published 1935)

Tate, D. M., Grier, R. A., Martin, C. A., Moses, F. L., & Sparrow, D. A. (2016). *A framework for evidence-based licensure of adaptive autonomous systems* (IDA Paper P-5325). Alexandria, VA: Institute for Defense Analyses.

Acknowledgments

Valuable comments reflected in the paper were provided by Phil Sarnecki. John Biddle, Nick Kaminsky, and Poornima Madhavan contributed valuable discussions.

ACQUISITION RESEARCH PROGRAM
GRADUATE SCHOOL OF BUSINESS & PUBLIC POLICY
NAVAL POSTGRADUATE SCHOOL
555 DYER ROAD, INGERSOLL HALL
MONTEREY, CA 93943

www.acquisitionresearch.net