INSTITUTE FOR DEFENSE ANALYSES

A Framework for Evidence-Based Licensure of Adaptive Autonomous Systems

David M. Tate
Rebecca A. Grier
Christopher A. Martin
Franklin L. Moses
David A. Sparrow

March 2016
Approved for public release;
distribution is unlimited.
IDA Paper P-5325
Log: H 16-000084

INSTITUTE FOR DEFENSE ANALYSES
4850 Mark Center Drive
Alexandria, Virginia 22311-1882

About This Publication

This work was conducted by the Institute for Defense Analyses (IDA) under contract HQ0034-14-D-0001, Project AK-2-3944, "Pedigree-Based Training and Licensure of Autonomous Systems," for the Air Force Research Laboratory (AFRL). The views, opinions, and findings should not be construed as representing the official position of either the Department of Defense or the sponsoring organization.

Acknowledgments

This paper has benefitted greatly from conversations with James R. Edmonson of Carnegie Mellon's Software Engineering Institute, Donald D. Davis of Georgia Tech Research Institute, and David H. Scheidt of John's Hopkins' Applied Physics Laboratory.

INSTITUTE FOR DEFENSE ANALYSES

IDA Paper P-5325

A Framework for Evidence-Based Licensure of Adaptive Autonomous Systems

David M. Tate
Rebecca A. Grier
Christopher A. Martin
Franklin L. Moses
David A. Sparrow

Executive Summary

IDA was tasked to assess potential benefits and challenges of applying an evidence-based licensing scheme to autonomous systems, and provide any insights gleaned to DoD. The autonomy community has identified significant challenges associated with test, evaluation verification and validation of autonomous systems. These challenges are exacerbated for adaptive and/or learning systems whose behavior on given inputs can change over time. The result is that a traditional comprehensive testing approach is impractical or impossible to implement. This report provides support for evidence-based licensure as a test, evaluation, verification, and validation (TEVV) framework that can address these challenges.

IDA found that traditional approaches to developmental and operational testing will not generally be sufficient to establish the safety and effectiveness of autonomous systems. A licensure approach does have the potential to support confident fielding of autonomous systems. To be successful, a licensure approach would require novel or modified activities throughout the life cycle of the system. These include:

- Explicit statement of safety and effectiveness requirements
- Formal methods during development
- Normative models of intended system behavior
- Modular and scalable virtual development test bed
- Run-time monitoring
- Periodic recertification of learning systems

Existing software and systems engineering techniques, if rigorously applied, provide a good start. Furthermore, there are ongoing research programs in most of these novel areas. That said, the approach must be supported by S&T investments aimed at overcoming distinct challenges of adaptive or learning autonomous systems. Candidate investment areas include:

- Mapping high level natural language requirements to testable (preferably machine testable) specifications
- Design of architectures that treat development and verification of capabilities on an equal footing
- Post-fielding testing of adaptive/learning systems

- Run-time monitoring and multiple-agent systems
- Coactive Design of human-machine teaming approaches
- Communication and Interaction among autonomous systems and between autonomous systems and humans.

Contents

1. Introduction 1
A. Background 1
B. Purpose 2
C. Planned Companion Volume....................3
2. A Candidate Licensure Approach for Autonomy....................4
A. Evidence-Based Licensure 5
1. Bounding the Confidence Space 6
2. Developing the Evidence—Oracle-Based Testing....................6
3. Arguments and Conditional Confidence 9
B. TEVV Framework....................10
3. Evidence-Based Licensure Life-Cycle Implications 12
A. Requirements....................12
B. Architecture 13
C. Design....................14
D. Development 14
E. Fielding....................16
F. Post-fielding 16
4. Summary and Conclusions....................18
Appendix A. Abbreviations 20

1. Introduction

A. Background

Adaptive autonomous systems have great potential to complement human performance in a wide range of missions. They show particular promise in contexts featuring high safety risk (either environmental or adversarial), very short response times, long endurance, tedious or repetitive activities, or difficult communications. To realize this promise, a number of difficulties must be overcome—not only in the science of adaptive autonomous systems, but also in our practical ability to develop, test, field, and operate such systems.

The Office of the Assistant Secretary of Defense for Research and Engineering (OASD/R&E) recently published a white paper identifying the goals, challenges, and principal areas of current technical shortfall with respect to developing, testing, and fielding autonomous and adaptive systems.[1] Here, we build on OASD's framework by extending the discussion of the challenges and proposing a specific schema for test, evaluation, verification, and validation (TEVV) of such systems. The schema we propose uses evidence-based certification and licensure to develop and maintain confidence in system dependability throughout the TEVV life cycle. In particular, we suggest that normative oracles—that is, top-down models of desired system behavior in various circumstances—can be used in a variety of ways to support TEVV of autonomous and adaptive systems for specific types of missions in a defined set of contexts.

In addition to describing a possible framework for TEVV, we suggest ways in which some existing software-assurance techniques could be adapted to work within it. We also examine the gaps in current TEVV capabilities, where new methodologies would be needed to support an effective licensure schema. We pay particular attention to the unique challenges posed by adaptive systems that *learn*—those whose behavior on a given set of inputs may change over time, even after the system has been fielded. We argue that in practice there may be no "end of test phase" for adaptive autonomous systems, particularly those that learn.

Currently, several obstacles make it particularly challenging to establish high confidence that an adaptive autonomous system will perform *dependably*—that is, that the system will behave as intended while safely, securely, reliably, and effectively performing

[1] *Technology Investment Strategy 2015–2028*, Autonomy Community of Interest (COI); Test and Evaluation, Verification and Validation (TEVV) Working Group; May 2015.

the assigned missions. The Autonomy COI TEVV Working Group has characterized the most significant challenges to TEVV as follows:[2]

- State-Space Explosion
- Unpredictable Environments
- Emergent Behavior
- Human-Machine Communication

These challenges are not unique to autonomy or to learning systems, but they are exacerbated in both of those domains.

B. Purpose

The purpose of this report is to outline a proposed approach that addresses these and other challenges and to provide a framework for consistent and traceable autonomous system verification and validation. In keeping with the broader literature, we classify our proposed approach as *evidence-based licensure*. The case for licensure will be built from individual *dependability cases* establishing reasons to believe that the system will perform safely and effectively within given limits.[3] Licensure thus refers to a level of confidence in all aspects of system dependability that justifies fielding. Both licensure and certification of the individual dependability cases will typically be both mission and context restricted. What we are seeking, then, is an *evidence-based licensure* framework for asserting likely system-level dependability of adaptive autonomous systems. The *Pedigree-Based Training and Licensure* concept advocated by the Autonomy COI falls within this same rubric.

Compelling evidence that the autonomous systems envisioned by DoD will dependably perform their missions and also dependably be safe and secure in operation will be hard to develop, but not impossible in our view. We have described an approach below that attempts to minimize the challenges. This will be accomplished in part through careful identification and specification of the missions and contexts in which the system can be expected to be dependable. The proposed approach borrows heavily from other disciplines that recognize (and in some cases have made progress on) similar challenges relating to the safety and effectiveness of decision-making software and automation. We acknowledge that we cannot say a priori how to approach all steps of the process, nor how hard the remaining work will be. We expect that the practical difficulties will vary considerably depending on the particular autonomy application.

[2] Ibid.

[3] Dependability cases generalize the widely used concepts of "safety case" and "assurance case" to other dimensions of desired system behavior. See, e.g., Charles B. Weinstock, John B. Goodenough, and John J. Hudak, "Dependability Cases," Software Engineering Institute Technical Note CMU/SEI-2004-TN-016, May 2004. We discuss dependability cases in more detail in Section 2.

The purpose of the body of this paper is to describe the necessary structure of any evidence-based approach. This includes identifying and pointing to a number of issues whose detailed treatment is deferred to technical appendixes planned for addressing those elements of the approach.

C. Planned Companion Volume

Note that developing the draft for this proposed framework revealed the need for dedicated treatment of a number of technical areas. These are called out in the main body for treatment in the planned companion volume. A thorough treatment of all the identified areas is beyond the scope of this project. The companion volume is organized with the following sections:

A. Introduction

This appendix introduces the remaining appendixes in the companion volume.

B. Formal Methods

This appendix provides a review of formal methods techniques, their applicability to licensure and autonomy, and extensive references.

C. Requirements and Metrics

This appendix describes a process for defining requirements and associated metrics that supports evidence-based licensure (EBL) for autonomous systems.

D. Normative Oracle Generation

This appendix describes the attributes of a Normative Oracle that would support EBL for autonomous systems.

E. CoActive Design

This appendix describes co-active design, which focuses on the interdependence of the human and the machine performing a joint activity. We might consider this an extension of teaming among humans, but that would imply a high degree of machine sentience.

F. Implications of Learning Autonomous Systems for TEVV

This appendix provides a review of formal methods techniques, their applicability to licensure and autonomy, and extensive references.

G. Modeling and Simulation Considerations for Licensure of Autonomous Systems

This appendix addresses modeling and simulation’s role in licensure of autonomous systems. To do so, key drivers from EBL and Autonomy are first identified to form a foundation for analysis.

2. A Candidate Licensure Approach for Autonomy

A *dependability case* articulates and argues for a *dependability claim*—that is, a desired behavioral property of the system and the operational contexts in which it must hold. Following the literature, we use "dependability" as a generic term to refer to safety, reliability, cybersecurity, operational performance, or any other dimension of user requirements. As with all requirements, there is a flow-down from high-level dependability claims to lower level claims supporting the higher level claim. For example, the high-level claim "The system will not crash into obstacles" might be supported by lower level claims such as "The system will be able to detect man-made structures during flight."[4] Individual dependability claims must be established (*certified*) as part of the process of structuring arguments for licensure.

The evidence supporting the dependability claim typically includes both process-based evidence (how the system was developed) and product-based evidence (demonstrable attributes of the system). Product-based evidence, in turn, may consist of both formally demonstrable properties of the system and empirical results of testing. A dependability case uses the logical and statistical relationships among the disparate pieces of evidence in a manner that supports a confidence assessment for the dependability claim. In common parlance, a *license* is permission from some authority to engage in certain kinds of goal-driven activity in specified environments. For example, a driver's license entitles the licensee to operate certain types of vehicles on public roads for certain purposes. The constraints on the activity are an important part of the license—motorcycles or buses require separate licenses from two-axle vehicles, and a basic license does not permit operating a commercial limo service.

Licenses are typically based on established *evidence* that gives the licensing authority adequate assurance that the licensee will exercise the relevant permissions in an acceptable way. In the case of a driver's license, the relevant evidence may include the age of the applicant, successful completion of an accredited driver's education course, successful completion of a road test, previous license history, and various other factors. A license is not a claim that dependability has been proven beyond all doubt; it reflects a sufficient degree of assurance for the purposes at hand.

[4] In theoretical computer science, it can be shown that any dependability claim may be decomposed into *safety claims* (concerning what will not happen) and *liveness claims* (concerning what must eventually happen). This greatly reduces the potential complexity of modeling dependability cases in formal systems.

For autonomous systems, the analogous approach would be for the controlling authority to license particular systems to conduct a specified set of missions, in a specified set of environments, under a specified concept of operations. In many cases, the “systems” will consist of teamed humans and cyber-physical systems. Licensure will depend on confidence that the licensed teams will perform dependably within the specified parameters.

A. Evidence-Based Licensure

Over the past few decades there has been a marked shift in software assurance theory and practice that provides a precedent for TEVV of adaptive autonomous systems. Software assurance is shifting toward a paradigm using *evidence-based certification of dependability cases.* This began in the safety community, where purely process-oriented assurance methods did not seem adequate for increasingly complex safety-critical applications in aviation, medicine, and civil engineering. Since then, the concept has been expanded to address not merely safety, but other operational outcome dimensions such as security, reliability, operational effectiveness, and consistency. The concept has also been extended to refer not merely to software, but to the integrated software/hardware/human system. The term *dependability* is often used to collectively refer to those outcome dimensions of concern for a given system development. In 2007, the National Research Council Committee on Certifiably Dependable Software Systems recommended adopting evidence-based dependability certification for critical systems.[5]

In this framework, confidence in system dependability is established through a collection of *dependability cases*, each of which addresses one or more outcome dimensions of interest. Dependability cases are a natural generalization of the “safety case” and “assurance case” concepts that have long been standard practice for safety and security certification of software-controlled systems. A dependability case consists of

- An explicit dependability claim.
- Evidence concerning the system’s ability to satisfy the dependability claim.
- An argument relating the evidence to the dependability claim.

The next sections focus on a framework for developing a satisfactory collection of dependability cases as defined above. The dependability cases should be collected in such a way as to establish the boundaries of the confidence space. We discuss methods to develop the evidence and quantify confidence in the argument defining the dependability case. Specifically, Oracle-based testing is proposed as a method of evaluating a system’s

[5] Daniel Jackson, Martyn Thomas, and Lynette I. Millett, eds., *Software for Dependable Systems: Sufficient Evidence?* (Washington, DC: The National Academies Press, 2007).

behavior against an external standard. Such methods are then integrated with known software paradigms in a framework for application in development of autonomous systems.

1. Bounding the Confidence Space

An important function of developmental testing is to not only maximize operational confidence across the entire range of intended operational contexts but also identify which narrower contexts permit the most confidence. In general, a system will be licensed for an increasing range of operations over an expanding set of contexts as it is developed and tested, but it will be possible at times to license a wider range of missions over a restricted set of contexts. A system that is highly effective in a narrow set of situations is not only potentially useful in its own right but also can improve our understanding of which evidence is most relevant to various contexts.

As an example, consider an autonomous unmanned aerial vehicle (UAV) that develops its own reconnaissance flight paths, given a specified target of interest. During development, it may turn out that the greatest challenges to system dependability are associated with weather conditions—high winds degrade both the UAV’s ability to travel its intended path and its ability to accurately estimate its air and ground speeds. This leads to collision-avoidance problems in environments with large vertical obstacles such as trees, buildings, or utility poles. For such a system, a natural progression of licenses might see the system certified first for self-directed reconnaissance operations in light winds over flat terrain, then separately for either heavier winds or obstructed terrain (but not both), then finally for general operations.

Similarly, systems that involve human-machine teaming may support multiple operating modes for teaming operations. Close control or teleoperation could be at one end of the spectrum of available modes, with full machine autonomy at the other. Close control will feature various patterns of human command, overrides, and mission updates while more autonomous operations will feature various patterns of status reporting, information requests, suggestions, and volunteered information, with or without machine learning. In practice, it is unlikely that all of these modes would achieve licensure simultaneously. The dependability of the system will be easier to establish for some modes than for others. Ideally, successful licensure and operations in easier modes can contribute to confidence in the dependability of more complex modes and perhaps eventually to adaptive selection of modes at run time.

2. Developing the Evidence—Oracle-Based Testing

In the software testing community, the term *oracle* has a variety of meanings. In some contexts, an oracle is simply a rule for stating whether or not a system has passed a given test. In others, it means any available information about the intended behavior of the system that could be used to evaluate performance. The common feature of all these meanings is that an oracle enables evaluation of a system's behavior against an external standard for what that behavior should be. Oracle-based testing (OBT) is thus any test methodology that explicitly depends on comparing observed (or instrumented) system behavior against an independent external standard of correct behavior. That comparison can result in a binary pass/fail evaluation or in a more nuanced scoring of degree of compliance.

At lower levels of hardware and software, OBT is similar to the use of *assertions* in software development. Software developers include statements in their code that make assertions about the current state—for example, that the value of a certain variable is within stated bounds, that a specific pointer is not NULL, etc. These statements are evaluated at run time, and if the assertion is not true, execution halts and the development environment enters a debugging mode. In OBT, the assertions might be about software states, hardware states (e.g., "voltage on this circuit within specified bounds"), or higher level behaviors (e.g., "air speed within 3% of target value"). When testing the system, these OBT assertions will similarly flag behaviors that meet expectations and signal the test community. To address the challenges of adaptive autonomous systems that were enumerated in the Section 1, we will need to be able to apply OBT at the complete system (or system-of-systems) level, including monitoring and assessing the human-machine interactions. We want to directly compare observed behavior of the system under test (SUT) against a functional description of desired high-level behavior.[6] To do this, we have to know what kind of behavior we are looking for and recognize when we do (or do not) see it. At the highest level of specification, this is little more than a statement of testable system requirements.

We will refer to a mechanism for assessing the desirability of a particular observed system behavior as a *normative oracle*. A normative oracle provides a process for scoring observed behavior against predetermined standards of desirability. That process has three distinct aspects:

1. A standard or specification of desirable and undesirable system behaviors.
2. A method for comparing observed system behaviors (including internals) against the standard.
3. A way of characterizing (and perhaps quantifying) the results of that comparison.

One can think of these loosely as "the answer key," "the grader," and "the score," respectively.

The most powerful normative oracles compare observed behavior against desired ideal behavior in a given circumstance, but this is not always necessary. Sometimes it will

[6] In this context, "observed behavior" may include the outputs of special test instrumentation that will not be part of the fielded design.

be sufficient to be able to score behavior in hindsight, without knowing in advance what the ideal response would have been. Normative oracles *do not make decisions;* they evaluate outcomes. It may be difficult to choose a good path through rough terrain; it is much easier for a normative oracle to know that getting stuck is a bad outcome.

Given a sufficiently rich set of normative oracles for the SUT, it is possible to implement a normative OBT framework for system development. In this framework, the normative oracles assess system behaviors against user requirements. They take as inputs measurements of system states and actions, as well as descriptions of the current situation and mission goals, and produce as output their evaluations of SUT behavior (and perhaps descriptions of what would be correct or preferred behavior in those situations). Normative oracles can be defined at high levels of description—for the system as a whole, for a major subsystem (e.g., for a path-planning module), for various cognitive functions, or for the human-machine teaming interaction. They can also be defined at very low levels, such as involving voltages, variable values, or instrument measurements. Essentially, any aspect of system behavior or performance for which it is possible to tell good from bad can have a normative oracle. In many cases, these oracles will be automated, but in others, they may have humans in the loop or be entirely defined by the subjective opinions of human experts.

Unlike system emulators and virtual environments, which are developed from the bottom up, normative oracles are developed from the top down, starting with high-level user requirements. For brevity, we will mostly give examples of system-level normative oracles, but we note that lower level oracles will also be essential for TEVV and can be used in series or in parallel.

A normative oracle provides a touchstone at every point during the TEVV life cycle for comparing what is happening against what ought to be happening. This, in turn, provides a basis for establishing evidence and quantifying confidence. As with human trainees, an important part of gaining confidence that true learning has occurred is seeing consistently appropriate responses to unforeseen situations. Conversely, even minor divergence between system behavior and the normative standard can provide valuable diagnostic information, even in cases where the system has “passed” the test at the macro level. Normative oracles can be particularly useful with respect to emergent behaviors of adaptive systems or systems of systems, in that they provide a framework for distinguishing desirable (or at least acceptable) emergent behaviors from unacceptable behaviors. In practice, normative oracles are needed not only for hardware and software performance but also for human operator performance, effective human-machine teaming, and machine learning.

Finally, sometimes normative oracles can be adapted to directly support operational dependability via run-time monitoring.[7] There is a natural process during development whereby the evaluations used by some oracles may be incorporated into the system (via a design change), becoming part of a self-monitoring and corrective-action feedback loop. For self-monitoring capabilities that get internalized (co-opted) into the system design, we no longer call the internalized test an "oracle"—it's now just part of a run-time monitor or control process. In such cases, the developers and testers will need to establish a standard for the desired behavior of the new feedback loop and devise new instrumentation and external oracles that monitor and evaluate the behavior of the new feature. Failure to do this can lead to situations like the Boeing 787 ADS-B bug, in which an oracle for (scoring whether transponder data packet boundaries were correctly aligned) was incorporated into an exception-handling routine (for misaligned packets)—but no new behavioral standard for the exception-handler was implemented, nor was its performance monitored during testing. As a result, developers were unaware of how frequently packets were being misaligned and how long it was taking the system to recover.

Not every oracle can or should be adapted in this way to yield an internalized self-monitoring capability. Automating the oracle's evaluation criteria in real time may not be possible at all, much less in a way that fits within the size, weight, and power constraints of the system under development. In addition, some oracles will always rely on human expert observation and judgment.

3. Arguments and Conditional Confidence

A core challenge for TEVV of adaptive autonomous systems is deciding when we are confident enough to license initial fielding for a particular mission in a particular context. A related challenge is reestablishing or enhancing confidence after the system has changed, be it through design changes, new operating concepts, or human or machine learning. How much testing, and of what kind, do we need to perform to establish the necessary confidence?

The problem, compared with TEVV of traditional systems, is our limited ability to model uncertainties associated with actual operating contexts. This is partly due to the processing and decision-making required of adaptive autonomous systems. Licensure for non-autonomous systems is based on assurance that the system will faithfully execute commands—either commands issued in real time or internal commands generated in response to the current system state. Vetting adaptive autonomous systems requires additional assurance that the system will dependably command itself—that is, it will make

[7] See, e.g., Kurt Rohloff, Joseph Loyall, and Richard Schantz, "Quality Measures for Embedded Systems and Their Application to Control and Certification," *ACM SIGBED Review – Special Issues on Workshop on Innovative Techniques for Certification of Embedded Systems* 3, issue 4 (October 2006): 58–62.

appropriate action decisions, in a timely manner, in a context that may involve interactions with other adaptive human or machine agents. How can such a system provide evidence that supports licensure?

Our TEVV approach is to identify a body of available evidence to certify that licensure for given missions and operating contexts is warranted with respect to particular dependability cases. Types of evidence include the following:

- Design for certification (formal methods).
- Performance on various tests.
- Comparison of system behavior (in test or in the field) against predicted and ideal behavior.
- Past certification of related dependability claims (composability).
- Details of the system's operational history in similar contexts.

Evidence might also plausibly include factors such as the track record of the developer or development methodology or previous successful uses of individual modules of the system (e.g., the motion planning engine or the learning algorithm). What constitutes useful evidence toward licensure will vary according to the type of system, operational contexts, envisioned missions, and degree of assurance required.

Successful dependability cases will require convincing arguments from demonstrable evidence, leading to a degree of confidence (preferably quantifiable) sufficient to justify certification. Historically, such cases have been constructed by hand, primarily in the areas of safety and security. There is ongoing theoretical work in the area of automated argument generation for both process-based and product-based evidence.[8]

B. TEVV Framework

Putting together the pieces described above, and with an eye on the specific TEVV challenges of autonomy, we can sketch the outline of a licensure-based TEVV framework for adaptive autonomous systems. Although here we use the language and paradigms of software testing, we emphasize that this framework applies to all phases and aspects of development for the complete hardware/software/human system. The framework has three primary elements:

1. A *normative model,* in which an exhaustive top-down set of derived normative oracles is implemented as executable code where possible.

[8] See, e.g., John Rushby, "Mechanized Support for Assurance Case Argumentation," in *New Frontiers in Artificial Intelligence*, Lecture Notes in Computer Science 8417, Springer, 2014.

2. An embedded *software testbed* that exercises actual system code in a virtual or partly emulated environment.
3. Actual physical instances of the SUT executing actual system code

At minimum, the normative model will assess and score observed system behaviors for desirability. Ideally, the model will also be able to specify the preferred or expected high-level system behaviors, given a description (at the appropriate level of detail) of the mission and context.[9] We emphasize that the normative model is *not* a simulation of the environment or of the actual system design under development, but instead is a reference or calibration of what characterizes “good” SUT behavior in a given situation, either in the testbed or in the actual environment. The normative model is a purely behavioral model—at high levels it will typically use idealized approximations of how the actual system *should* behave, without attempting to emulate detailed sensing/reasoning or real physics processing.

The purpose of the embedded software testbed is to allow progressive testing of system modules in isolation and in combination, working from unit verification and validation to full autonomous system-of-systems validation by stages. At every stage, some or all of the software and hardware modules of the SUT (and the human-machine interactions) may be emulated using the normative model, with the SUT providing the remaining functions. The testbed will provide simulation of the operational environment, including human commanders, other agents, and adversaries. Ideally, this testbed will have provisions for human-in-the-loop participation in both commander/operator and adversary roles, to provide realistic human-machine teaming. In some cases, the testbed could provide virtualization support for live virtual constructive testing.

The ultimate goal of this iterative testing regime is to be able to establish the licensability of specific system modules (or entire systems) based on arguments using composable dependability cases covering a specified range of contexts. Section 3 examines specific uses of the testbed, and associated challenges and capability gaps, associated with specific phases of the TEVV life cycle.

[9] We use the word “behavior” here in its broadest sense, to include both observable actions by the system and internal processing and all dimensions of performance and dependability. Scoring metrics for these various dimensions will include both continuous measures and binary (pass/fail) measures (see planned companion volume).

3. Evidence-Based Licensure Life-Cycle Implications

The consensus in the dependability literature seems to be that achieving dependable systems (with high confidence) will require a multifaceted approach. Formal design processes, normative models, empirical testing, run-time monitoring, and explicit limitations on operational parameters can all be used to support licensable confidence in adaptive autonomous systems. Applying these techniques will require special actions throughout the development and TEVV life cycles. In this section we give a brief overview of how EBL might affect the execution of various activities and phases.

A. Requirements

EBL is about generating confidence in system dependability, which is assessed against requirements. As a result, it is vital that the elicitation and specification of system requirements be fully integrated into the EBL framework from the start. To begin with, any normative oracle used in EBL must accurately reflect stakeholder preferences for system behavior—which is to say, requirements. To realize the benefits of normative oracles for EBL, requirements (including assumptions about concept of operations, environment, human-machine teaming, and interactions with other agents) must be captured and codified in a form that supports implementation of a normative model. Current practice does not typically yield formal system requirements that are sufficiently complete, unambiguous, and testable to support automated generation of normative oracles. The planned companion volume addresses issues associated with requirement specification and normative oracle generation.

Second, there is considerable ongoing research into formal analytical models to generate provably dependable code.[10] Such methods necessarily begin with a description of the behavioral requirements (or prohibitions) to be proven. If "design for dependability" development methods are to be used, the provable certifications for the resulting design will be contingent on that original requirement specification. The planned companion volume addresses the challenges of incorporating formal methods into development.

Finally, the problem of empirical testability will be particularly acute for adaptive autonomous systems. It will not be enough for requirements to be complete and formally specifiable; those behavioral requirements that are not a priori provable must be testable in

[10] See, e.g., V. Janarthanan and A. Gherbi, "Architectures for Dependable Software Systems," *Proceedings of the Eighth International Conference on Information Technology: New Generations*, IEEE, 2011.

practice. Deriving statements of high-level behavioral goals that can be put into practice is challenging even for non-autonomous systems; autonomy and learning add a new layer of complexity. Specifying requirements for how systems should reason and learn and how they should interact with others, and devising metrics and oracles that make those requirements testable, is a key challenge. During developmental testing, automated tools for deriving testable assertions (at the system-state level) from higher level behavioral requirement descriptions would be useful not only for test design but also as a diagnostic tool and in support of low-level normative oracle implementation. The planned companion volume deals with the particular challenges of learning systems for TEVV.

B. Architecture

Formal methods for generating validated code (and test cases) directly from requirements offer great promise in contributing to dependability arguments. If such methods are used, they must be selected and committed to from the start, during the architecture phase of system development. Hardware architecture, software architecture, processing architecture, test architecture, and the interfaces among them must be chosen to support the formal methods that are to be used. Any subsequent changes in the architectures (e.g., in response to requirements changes or resource constraints) run the risk of invalidating dependability arguments that depend on formal properties and processes.

At the same time, the architecture(s) selected may have significant implications on testability. Balancing the demands of formal provability and empirical testability calls for early and careful consideration of the relative demands of both, along with the relative contributions that each will be able to make to the planned dependability arguments for the system. The reasoning module's architecture must also balance considerations of capability (i.e., extent and flexibility of potential learning) against the ability to certify the various dimensions of dependability. A reasoning architecture that supports a tiered implementation of learning, with higher tiers supporting more sophisticated behaviors that are more difficult to license, could also support iterative and composable licensure—but only if such an architecture is adopted early in system development. Similarly, run-time monitoring of reasoning functions may be indispensable for licensure, but must be incorporated into the architecture from the start.

There is already a literature on architecture implications for safety.[11] Extending current safety models to address general dependability cases seems straightforward in principle, but will almost surely be less straightforward in practice. This is an open research area.

[11] See, e.g., Weihang Wu and Tim Kelly, "Towards Evidence-Based Architectural Design for Safety-Critical Software Applications," in *Architecting Dependable Systems IV*, ed. Rogério de Lemos, Cristina Gacek, and Alexander Romanovsky (Berlin, Heidelberg: Springer-Verlag, 2007), 383–408.

C. Design

"Design for dependability" is a natural extension of the "Design for safety" approaches to safety-critical systems. As with architecture, there is a premium on early systems engineering to take maximum advantage of reliable processes and model-based methods during development. This will be especially true for adaptive autonomous systems and their teaming with humans.

Several of the more promising approaches to dependability rely on explicit system modeling of dependability properties (e.g., safety, security, survivability, effectiveness) combined with run-time monitoring of dependability implications by independent on-board processes (e.g., normative oracles).[12] Adding such "introspective AI" features late in development to a system whose original design did not incorporate them organically could greatly increase the difficulty both of achieving good performance and of establishing confidence in dependability.

As with architecture (again) there will be a tension between the potential capability of a design and the testability of that design. An explicit goal of the EBL approach is to make this tension explicit and quantifiable and to support incremental licensure of restricted applications of a given design. Absent a design approach that explicitly facilitates testing, the resources required for testing will grow rapidly as a function of system capability.

D. Development

Once requirements have been elicited, normative oracles have been implemented, an architecture has been selected, and a design has been proposed, the actual development of the system can productively begin. Simultaneously, this is when iterative certification of dependability cases begins.

There is a great deal of work still to be done in the theory and practice of exactly how evidence acquired during development can most effectively be organized into successful arguments for licensure within specified bounds. As with most aspects of dependability, work on safety assurance is currently the most advanced, but even that literature offers little in the way of concrete mechanisms for generating arguments, composing module-level licenses into system-level licenses, or designing efficient tests to support licensure. Promising directions have been identified; it remains to translate those into practical development tools and to expand them from the domain of safety alone to other dependability dimensions.[13] In particular, we suspect that formally generated normative

[12] See, e.g., part 3 of John Fox and Subrata Das, *Safe and Sound: Artificial Intelligence in Hazardous Applications* (Cambridge, MA, and London, England: AAAI Press / MIT Press, 2000).

[13] Rushby, "Mechanized Support for Assurance Case Argumentation." See also Rajwinder Kaur Panesar-Walawege, Mehrdad Sabetzadeh, Lionel Briand, and Thierry Coq, "Characterizing the Chain of

oracles will play a substantial role in the design, development, and licensure of adaptive autonomous systems.

For the specific challenge of architecting, designing, and implementing effective interactions between human and machine agents in a teaming context, recent work on Coactive Design seems particularly well suited to development in an EBL framework:

> Coactive Design is a new approach to address the increasingly sophisticated roles that people and robots play as the use of robots expands into new, complex domains. The approach is motivated by the desire for robots to perform less like teleoperated tools or independent automatons and more like interdependent teammates.[14]

In particular, Coactive Design asserts that the appropriate degree and type of autonomy for a given system is the autonomy that will produce the most effective teaming behaviors, in terms of mission performance. Expanding this focus from mission performance to dependability is straightforward. In general, the most effective teaming relationship will feature neither the least machine autonomy nor the most autonomy. Rather, it will exhibit human-machine interactions in which agents issue commands and request or volunteer information in ways that best support dependability.

Another key research area will be how to respond to requirements changes during development. Requirements changes pose a significant challenge even for “normal” systems development. For adaptive autonomous systems being developed under an EBL framework, requirements changes simultaneously disrupt both the design process (i.e., making the system work properly) and the licensure process. There is a very real danger that design changes driven by new or modified requirements could invalidate the arguments underlying existing dependability case certifications. Any licenses that rely on those certifications would be undermined. There is thus a “double effort” associated with engineering change proposals—one effort to accomplish the change and a second effort to re-license the changed design, possibly with different context limitations. A strong theory of composability of evidence would be helpful—both to minimize the amount of licensure re-work and to support design of regression tests for dependability.

Finally, note that it will also be necessary to test and validate the EBL testbed itself. Normative oracles, simulation models (both of the environment and yet-to-be-implemented modules of the SUT), human-in-the-loop interfaces, and the set of dependability cases are

Evidence for Software Safety Cases: A Conceptual Model Based on the IEC 61508 Standard,” *Third International Conference on Software Testing, Verification and Validation*, 2010, 335–44, and Sunil Nair, Jose Luis de la Vara, Mehrdad Sabetzadeh, and Lionel Briand, “An Extended Systematic Literature Review on Provision of Evidence for Safety Certification,” *Information and Software Technology* 54 (2014): 689–717.

[14] Matthew Johnson, Jeffrey M. Bradshaw, Paul J. Feltovich, Catholijn M. Jonker, M. Birna van Riemsdijk, and Maarten Sierhuis, “Coactive Design: Designing Support for Interdependence in Joint Activity,” *Journal of Human-Robot Interaction* 3, no. 1 (2014): 43–69.

all subject to errors and omissions. Debugging the debugger and verifying the verification protocol will be nontrivial efforts. For this reason, there is great potential value in the development and licensure of reusable test infrastructure—high-level dependability case suites, automated oracle generators, and virtual environments for EBL.

E. Fielding

An important part of the development process will be methods for identifying the operational contexts within which confidence in the current system is highest. Progressive licensure during development can be based either on showing that system capabilities have improved or by identifying operational constraints for which the current system's behavior is particularly dependable.

Fielding can occur when the system is licensed for a subset of missions and context bounds that provide threshold operational utility. For example, an unmanned aerial system intended eventually for day/night, all-weather, multi-sensor intelligence, surveillance, and reconnaissance missions might initially be licensed only for electro-optical/infrared imaging for daytime, fair-weather, low-wind conditions. If that restricted capability has sufficient operational utility, limited fielding is justified.[15] In this case, it is important that data from the operational experience be leveraged to provide additional evidence in support of wider licensure. Integration of information from the field with ongoing developmental testing will be the norm, rather than the exception, requiring new and better coordination of test organizations and data systems.

F. Post-fielding

The ongoing expansion of licensure even after initial fielding blurs the traditional line between "development" and "operations." Data from the field can inform licensure of unfielded systems; data from test events can inform licensure of fielded units. In a very real sense, the developmental testing of the system does not (and should not) end.

This will be especially true of systems capable of learning. Because the system's behavior will be changing over time, as will its teaming capabilities, it will be necessary to perform regression testing of system dependability to re-license the new configuration. This might result in a wider range of licensed missions and contexts or an adjusted range. (If the range narrows, it may be possible to reset the system to its original fielded state.) These tests might be periodic, or event-driven, or both. Run-time monitoring of reasoning

[15] This "threshold" might not be the threshold capability originally identified during requirements elicitation. In this respect, EBL resembles agile software development—the first combination of capability and dependability that is sufficient for fielding is something that the developers, testers, and users will "know when they see it."

functions might spot an increased mismatch between normative oracles and system decisions, triggering a “recall” of the unit.

Even for systems without adaptive learning, there will undoubtedly be upgrades and new features over time. Re-licensure of these systems will also involve regression testing of dependability. Design of time- and cost-effective regression tests (and the corresponding licensure arguments) is thus an important research area for adaptive autonomous system TEVV.

4. Summary and Conclusions

TEVV of autonomous systems poses a number of distinct challenges. Adaptive and/or learning autonomous systems, resulting in the possibility of the same input leading to different behaviors, complicates test and evaluation. In addition, the impact of dynamic environments and an effectively inexhaustible state-space for the system are not unique to autonomous systems, but are more challenging to test and evaluate in the presence of autonomy. Interactions between autonomous systems and their operators (or adversaries) are also additionally complex.

This paper addresses these challenges using a licensure paradigm. We enumerate and discuss the necessary features for a TEVV paradigm that would permit licensure of adaptive autonomous systems analogous to the way we license human drivers to operate specified classes of vehicle in specified conditions for specified purposes.

Fortunately, we can capitalize on existing work on complex software systems—especially those needing safety or security certification in cases of dynamic environments and inexhaustible state spaces. Considerable progress, both theoretical and practical, has been made in such fields as aviation safety, medical hardware/software systems, and space exploration. This body of work informs and is consistent with the EBL paradigm we describe. EBL provides a mechanism for establishing probabilistic confidence in system dependability and for determining operational limits on the license in terms of missions and contexts. We can also expand the basic EBL framework through the use of normative oracles, in particular normative models that implement executable specifications of desirable system behavior. Normative models can be useful both as testing tools and as contributors to run-time dependability.

While capitalizing on this body of work is probably essential for going forward, the progress to date in TEVV is insufficient to support fielding of autonomous systems that have learning capability. Gaps in established practice—and in some cases in extant theory—have been identified throughout the TEVV life cycle. At the start of development, better ways to automate (or machine assist) the translation of high-level system requirements into lower level behavior specifications on subsystems, and the translation of natural-language behavioral specifications into implementable normative models, are needed. When system architecture is developed, guidelines must be established for choosing architectures that will support formal derivation of system dependability where possible and testability of requirements that will need to be empirically verified. During development, the TEVV community will need improved understanding (and automated support) for assessing the degree of confidence that is justified by the formal attributes, observed test performance, and other accrued evidence for system dependability—especially if there have been changes to system requirements since the original architecture and normative model(s) were developed. At the "end" of development, fielding will require ongoing monitoring and test and evaluation—especially for systems that can learn—and will thus not necessarily continue to behave as they did during formal development test.

More detailed treatments of all of these issues (and others) are required before the EBL testbed we propose can be implemented in practice. The planned companion volume to this document provides a first cut at enumerating and beginning to address some of these concrete implementation issues. Identification of the critical science and technology research areas relevant to evidence-based licensure will be an essential prerequisite for implementation.

Appendix A.
Abbreviations

COI	community of interest
EBL	evidence-based licensure
OASD/R&E	Office of the Assistant Secretary of Defense for Research and Engineering
OBT	oracle-based testing
SUT	system under test
TEVV	test, evaluation, verification, and validation
UAV	unmanned aerial vehicle

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE March 2016	2. REPORT TYPE Final	3. DATES COVERED *(From–To)* Jan 2016 – Mar 2016

4. TITLE AND SUBTITLE A Framework for Evidence-Based Licensure of Adaptive Autonomous Systems	5a. CONTRACT NUMBER HQ0034-14-D-0001
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Tate, David M. Grier, Rebecca A. Martin, Christopher A. Moses, Franklin L. Sparrow, David A.	5d. PROJECT NUMBER AK-2-3944
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Institute for Defense Analyses 4850 Mark Center Drive Alexandria, VA 22311-1882	8. PERFORMING ORGANIZATION REPORT NUMBER IDA Paper P-5325 Log: H 16-000084
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory 2610 7th Street, Bldg 441 Wright-Patterson AFB OH 45433	10. SPONSOR/MONITOR'S ACRONYM(S) AFRL
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited (17 May 2016).

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Adaptive autonomous systems of interest to DoD have great potential to complement human performance in a wide range of missions. This particularly is true for adaptive systems that learn—those whose behavior on a given set of inputs may change over time, even after the system has been fielded. Such adaptation, however, makes exhaustive testing, certification, and licensure of the final system impossible. The challenge is to establish high confidence that the system will perform dependably and behave as intended while safely, securely, reliably, and effectively carrying out the assigned missions. This paper adapts approaches from other disciplines, such as software assurance theory, where safety as well as performance is paramount. The key features of the approach include taking and retaining performance data from the beginning of development, establishing a formal means to assess actual performance compared with desired performance throughout the development, and recognizing that system testing will need to continue beyond a fielding decision. An additional complexity is that performance testing needs to address how the autonomous modules make decisions and provide a standard for doing that. That is, test, evaluation, verification, and validation of such systems must inform testers about how a system satisfies requirements during development and about its potential to make effective system changes after fielding.

15. SUBJECT TERMS

autonomy, TEV&V, adaptation, Normative Oracles, licensure, certification, dependability cases

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON Ms. Kristen Kearns
a. REPORT Uncl.	b. ABSTRACT Uncl.	c. THIS PAGE Uncl.	SAR	27	19b. TELEPHONE NUMBER *(include area code)* (937) 656-9758