Carnegie Mellon University
Software Engineering Institute

CENTER FOR CALIBRATED TRUST MEASUREMENT AND EVALUATION (CATE)—GUIDEBOOK FOR THE DEVELOPMENT AND TEVV OF LAWS TO PROMOTE TRUSTWORTHINESS

Andrew O. Mellinger, Tyler Brooks, Chris Fairfax, Daniel Justice

April 2025

DOI: 10.1184/R1/28701104

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Table of Contents

1 CaTE Guidebook Overview

The purpose of this guidebook is to provide operational test and evaluation (OT&E) and developmental test and evaluation (DT&E) personnel with observations and recommendations that help them effectively develop, test, evaluate, verify, and validate (TEVV) lethal autonomous weapons systems (LAWS) that function with the use of machine learning (ML) models, specifically with regards to the ideas of system trustworthiness and operator trust in these system.

This guidebook is a product of the Center for Calibrated Trust Measurement and Evaluation (CaTE), which is a program that was jointly chartered by the Office of the Under Secretary of Defense for Research and Engineering (OUSD(R&E)) and the Carnegie Mellon University Software Engineering Institute (CMU SEI). The main purpose of CaTE is to establish methods for evaluating operator trust and to assure the trustworthiness of artificial intelligence (AI) systems. CaTE aims to "[address] fundamental complexities and engineering challenges associated with AI assurance, and developing some of the standard framework methods and processes for how the Pentagon evaluates trustworthiness" [Harper 2023].

The need for this guidebook arises out of Department of Defense (DoD) Directive 3000.09, which was updated in January of 2023 to allow for "developing and using autonomous and semi-autonomous functions in weapon systems" up to and including those that use lethal force [DoDD 3000.09]. Development of such autonomy often entails incorporating AI into these systems, so the directive seeks to provide policy and guidance around the use of AI. To that end, this guidebook aims to support the following objectives from DoD Directive 3000.09:

- understand and evaluate operator trust
- provide evidence about a trustworthy system
- establish how to create a trustworthy system that also promotes operator trust
- promote responsible AI (RAI) principles and tenets

1.1 Rationale and Background: Trustworthiness in ML

On May 26. 2021, the DoD issued the memorandum "Implementing Responsible Artificial Intelligence in the Department of Defense." This memorandum outlined the five DoD AI ethical principles, which dictate that AI should be responsible, equitable, traceable, reliable, and governable. In addition, it defines the following six foundational tenets: RAI governance, warfighter trust, AI product and acquisition lifecycle, requirements validation, RAI ecosystem, and AI workforce. All the principles and tenets promote the establishment of a responsible set of processes for developing a trustworthy and trustable system. However, we want to emphasize the second tenant on warfighter trust, which is as follows: "Ensure warfighter trust by providing education and training, establishing a test and evaluation and verification and validation framework that integrates real-time monitoring, algorithm confidence metrics, and user feedback to ensure trusted and trustworthy Al capabilities" [DoD 2021a].

Although this guidebook supports all the principles and tenets throughout all the stages of the product lifecycle, the core goal of the CaTE program is to promote trust. We therefore quote the "warfighter trust" tenet to emphasize the leading aims of this guidebook. Trust is essential because any operator that doubts the behavior of any machine might be distracted from his or her focus, and the most minimal of distractions can significantly decrease the effectiveness of individual operators as well as entire teams.

To aid in the TEVV of LAWS while making the tenet of trust central, this guidebook provides recommendations and observations on how to discuss, design, and integrate trust, trustworthiness, calibrated trust, and ethics based on emerging practices, frameworks, metrics, measures, formats, and tools. We provide direct contributions toward the understanding of trust, trustworthiness, and ethics and provide specific ways of evaluating them with respect to LAWS. The goal of CaTE is not to reinvent system or software engineering but to build on strong, existing foundations while incorporating the latest and greatest innovations from current investment. However, current standards are few, and the ones that exist tend to focus on non-lethal autonomy in the public sector, such as self-driving cars. These standards don't necessarily account for the complexities covered in DoDD 3000.09, and there is currently no centralized standard that precisely targets CaTE's scope.

To achieve trustworthiness, we must address numerous issues in several key areas. The use of AI, and in particular ML, introduces new and significant challenges in the development and testing of mobile LAWS, especially ML that affects decision-making processes involved in navigation, perception, targeting, and response. Semi-autonomous fixed weapons systems such as the close-in weapon systems (CIWS)[1] have been in use for decades, and the engineering and testing of these systems is well understood. However, ML introduces many additional complexities such as a dependency on data or training environments, ambiguity in outputs, increased system demands, and vulnerability to deception, among many others. To address these issues in detail, the recommendations we provide in this guidebook assume the use of the highest complexity devices—such as small, form factor quadrupeds—operating in the most challenging environments—such as urban conflicts with a mix of civilians and warfighters.

In addition to these complexities that determine whether ML works as technically intended, the use of ML in LAWS and its ability to make decisions also has ethical implications—another enormous consideration for system trustworthiness and promoting user trust. Ethics is a complex and nuanced subject, and it is often situationally dependent and based on dynamic and often subtle factors. For this reason, it is critical that commanders and operators understand and trust that the machine's behavior is governed by a well-understood and clearly communicated set of ethical rules or models. Therefore,

[1] For a short definition and overview of close-in weapon systems, see the following: https://en.wikipedia.org/wiki/Close-in_weapon_system

systems and design rules demand significant effort and clear engineering requirements during development to meaningfully test and verify them later.

We support using the idea of *calibrated trust* to guide developers in making decisions about specific environments and conditions when building these systems. Calibrated trust provides a behavior profile for the machine based on its performance in specific environments and situations so that users can have a high level of trust in its behavior, rather than relegating trust to a baseless feeling. In addition, to support ethical behavior, we introduce the concept of *conduct assurance***.** The National Institute of Standards and Technology (NIST) defines assurance as follows: “Grounds for justified confidence that a claim has been or will be achieved” [NIST 2024]. We therefore emphasize the importance of defining ethical rules to help generate trust by establishing the conduct that the machine must follow, and, throughout this guidebook, we propose a variety of ways that conduct assurance case claims can be supported with practices and artifacts throughout the development and testing process.

The significant challenges we’ve outlined here in addition to the constant flux in ML technologies necessitate frequent reevaluation of testing techniques, tools, and practices. These challenges and the constantly changing landscape hinder the identification of best practices and consequently the standardization of approaches, as many of them have yet to exist. We’ve tried to create a complementary set of practices we think provide a body of evidence that is sufficient to TEVV these products. However, to continue supporting the changing landscape of ML, we intend to update this guidebook on a regular basis to provide better recommendations and observations about the current state-of-the-art and state-of-the-practice techniques as they continue to evolve with the technology. There are many unsolved challenges, and sections of this guidebook provide “best effort” solutions, guidance for how to fill gaps, expectations for the future, and calls for future work until better options are available.

1.2 Focus Areas

The following sections provide an overview and explanation of how we developed this guidebook, and how its structure can help readers understand what it provides. The CaTE pilot focused on four areas during its development, which we outline in these sections to explain how we address each area.

1.2.1 Science and Technologies

This area addresses the fundamental basics of AI and ML. Because AI and ML techniques and tools change on a regular basis, this focus area tries to answer questions about how model capability and performance change over time. It looks at topics like measures and metrics of individual model performance, measures and metrics of model ensembles, model reliability, robustness and calibration, and how these issues affect trustworthiness and operator trust. Ultimately, CaTE is interested in how the end systems perform, but such assessment isn’t possible without understanding their base technologies.

1.2.2 Engineering Tools and Practices

This area addresses practices, frameworks, processes, formats, and tools that can help produce a body of evidence in support of an assurance case towards trust, trustworthiness, and ethical behavior.

Interest in AI has increased rapidly in recent years, resulting in staggering growth in investment. According to a recent article in *Crunchbase News*, AI startup funding alone exceeded $100 billion in 2024 [Teare 2025], dwarfing the DoD budget of $1.8 billion for 2025 [Vincent 2024]. This funding has led to enormous growth in the tools available for AI, and it does not even directly include autonomous systems. In such a quickly changing environment, we aim to recommend some reasonable starting points for adopting new and useful tools for important processes and approaches to ML development and testing.

1.2.3 Human-Machine Teaming (HMT), Tools, and Practices Evaluation Lab

We can't understand the effectiveness of science and technology practices until they are applied. This area emphasizes the importance of having a hands-on environment where we examine tools and practices to evaluate and measure how they support the trustworthiness case and operator trust. We seek to answer questions such as the following:

- How do tools and practices reveal important information about the development process and support observability, governability and equitability?
- How effective are they at achieving desired metrics?
- How easy are they to learn?
- How easy are they to use?
- How easy are they to adapt to new machines?

1.2.4 Workforce Development

This area promulgates the best practices and advancements in the states of the art and practice to deliver them quickly and effectively into the hands of our users. This area focuses on developing and transitioning the most effective and meaningful training in the areas of foundational AI engineering and the effective TEVV of LAWS to guide the adoption and use of the practices and tools we evaluate.

1.3 Scoping: AI Versus ML

There are many definitions of AI currently in use even within the DoD. The Defense Acquisition University (DAU) has provided a "DoD AI Definition Reference Chart" that attempts to reconcile and align three different definitions. In general, these definitions have similar goals in that they stress how the AI is used to recognize patterns, make recommendations, and support decision-making in ways that normally require human intelligence [DAU 2024].

There are many ways to provide the capabilities outlined in these definitions of AI, such as through expert systems, fuzzy logic, and ML, to name a few. Currently, most technological advancement focuses on ML over these other methods and, more specifically within ML, on deep neural nets (DNNs). For these reasons, we focus on the subset of ML technologies within the larger field of AI throughout this guidebook.

ML approaches bring significant changes to existing engineering practices. Consequently, CaTE's focus is on ML-enabled systems [Horneman 2019]. This version of the guidebook addresses supervised learning, and we therefore leave reinforcement learning, and its challenges, to subsequent versions.

1.4 How to Read This Guidebook

The primary intended audience for this guidebook is the personnel performing the development and operational TEVV of DoD LAWS for which adherence to the principles of RAI and DoDD 3000.09 are required. Such individuals will benefit from reading this guidebook in its entirety.

Related audiences, such as AI, ethics, and society developers and acquisition personnel, will also find this guidebook of significant benefit to help them understand how to develop and evaluate trustworthy systems. All other audiences are likely to get the most value by reading the Overview chapter and the observations and challenges sections of ensuing chapters.

The main chapters in this book address what we believe to be most important for the TEVV of LAWS. The book is not intended to comprise an exhaustive list of all TEVV concerns, but the topics we cover will serve to guide audiences through their most pressing issues. The first four chapters cover broad issues and cross-cutting concerns such as trust, ethics, and software development life cycles (SDLCs). The subsequent chapters provide guidance at various stages of the SDLC, focusing on how the introduction of ML impacts LAWS. We suggest reading the first four chapters entirely, and then the remaining as needed.

Each of the subsequent chapters contains an overview of what the chapter covers, guidance on the topic, observations our team has made, challenges personnel will face, future work we expect will be needed on this topic, and, finally, details to provide background for the previous sections.

1.5 Key Findings

AI, ML, and autonomy are not new, but they are undergoing rapid evolution with many contributors across academia, industry and government. Such rapid evolution results in the addition of capabilities at a tremendous pace to the both the state of the art and state of the practice. Due to this constant change and rush to market, we have yet to reach a consistency or a sufficient level of maturity from which to establish best practices and techniques that could provide an engineering rigor for safety critical systems such as LAWS. The fields of AI and robotics both contribute to these issues as well as to other technical, process, and cultural challenges in the TEVV of LAWS [Afzal 2020; NAS 2023; Rismani 2023]. In this guidebook, we provide guidance for many of these areas, and the following sections provide brief summaries of the findings and themes that came out of this work.

1.5.1 Defining Systems (From Models to Systems)

Testing is based on the premise of knowing what the right answer is. In systems, “the right answer” comes from real mission needs, operators, and practical experience. However, finding those real examples has been difficult because fully autonomous, squad-level weapons systems are rare. We were therefore unable to find publicly available examples or guidelines for writing good AI or ML requirements, especially for LAWS.

Establishing guidelines to generate consistent goals, language, measures, metrics, and techniques for specifying AI, ML, and LAWS requirements would significantly improve the ability to test these systems. Similarity in system specifications, even for different systems, enables the transfer of knowledge and the extension of approaches to other systems. We explore this area in Chapter 5 and provide example guidelines.

1.5.2 The Need for Concreteness

The sources we reviewed offer sound, high-level guidance, but they lack concrete specifics. While such generality is understandable given rapid technological advances, it still places significant burden and uncertainty on individual projects. As the field evolves, we anticipate that publications will move beyond general questions like “Does the system have good transparency?” and toward measurable evaluation frameworks like those offered by IEEE 7000-2021, which provides ordinal transparency levels for autonomous systems [IEEE 7000-2021].

1.5.3 System Scope: Working from Small to Large

Most of the sources we consulted focus on single models or small sets of similar models. A few sources, such as those that deal with self-driving cars, address large systems of models with complex integration logic. As we move into more complex, multi-model, ML-enabled systems—and especially LAWS—we need to emphasize and capture how we specify, design, and test advanced hybrid systems to help develop the LAWS engineering discipline.

1.5.4 Conflating AI and ML

The terms AI and ML are often used together or interchangeably, though not all AI technologies are learning systems like expert systems. ML models typically generate predictions about inputs (e.g., classifications, object identifications, or next-word predictions) that must then be integrated with decision-making logic to determine actions. From an integration perspective, a complete AI system combines both the ML prediction component and this decision-making logic.

Many sources we reviewed were ambiguous about which part or combination they addressed. When discussing “AI accuracy,” sources were often unclear about whether they were referring to the ML model’s prediction accuracy or the accuracy of decisions the system made using those predictions. For effective development and testing of ML-enabled systems, identifying this transition point enables component isolation, observability, and targeted testing. We emphasize this critical distinction throughout our guidebook and supporting documentation.

1.5.5 Continuous Learning

Sources we consulted often assume that ML-enabled systems are *continuous learning* systems in which the AI or ML components are constantly changing over time based on direct mission experience. This assumption leads to both inflated expectations about capabilities and concerns about vulnerabilities, all of which contributes to the fundamental challenges involved in developing operator trust. We need to provide a clearer and more realistic picture of what the capabilities are. In the future, we need to clarify to design and testing teams when and how learning will occur—whether learning happens during development, periodically during deployment through explicit updates, or through on-platform tuning on a frequency that we need to specify.

1.5.6 Trust, Ethics, and Conflict

Trust, trustworthiness, and ethics in AI and autonomous systems are receiving a tremendous amount of attention. The new capabilities provided by AI and ML introduce new areas of automated decision-making that challenge our understanding of ethics. Therefore, we must design the devices we make to act in an ethical way. Ethics covers a broad set of moral principles, although some definitions narrow the field to directly address the safety, health, and welfare of the public and goal of improving society [Institute for Ethical AI & Machine Learning 2025; IEEE 2025].

Because of the nature of ethics, and because people are quick to disagree on ethical principles, it becomes difficult to consensually arrive at a single instance of ethics when designing and engineering a system, especially with a system that can apply lethal force. Therefore, we have found that separating the definition of specific ethical values from designing a rational engineering process is critical to enable the development of the supporting technology. However, even when separate, we must work to codevelop each area in parallel while paying attention to their dependent relationship.

1.5.7 Bootstrapping the Operationally Relevant Tests

Developing operationally relevant tests for LAWS faces two major challenges. First, we lack knowledge about these systems' real-world applications, about their functionality in the field, and about how they interact with operators. Without this understanding, the "train as you fight" philosophy relies on speculation. This lack of understanding is common with disruptive technologies, but LAWS introduce so many novel capabilities that the operational shift will be both significant and more difficult to anticipate than usual for disruptive technologies. Second, user testing based on simulations or prototypes cannot provide sufficiently realistic environments for effective testing, especially with enough "perceived risk" to gather accurate user feedback. When studying warfighter trust, we must address these challenges by emphasizing realistic test environments, introducing prototypes early, and engaging warfighters throughout the development process.

1.5.8 Ops Cycles and Deployment

Most of the commercial ML model providers emphasize rapid updates, cutting-edge techniques for evaluation, and fast delivery to market for new use cases that do not pose critical safety concerns.

MLOps, which has emerged from DevOps, embodies continuous development and deployment. The rapid develop-deploy cadence of MLOps may not be appropriate for a safety-critical LAWS and may even increase the cost of LAWS-development efforts. The ops paradigm provides strong support for a reliable and rapid development-to-deployment process but requires extra cost to achieve full automation that may be unnecessary and even prohibitive for LAWS. As a result, many of the commercial technologies, while attractive, may not directly meet the special development and testing needs of LAWS. It is unclear at this time what the preferred ML-enabled SDLC will be for acquisition.

2 Trust, Ethics, and Human Systems Integration

As we established in the discussion in Section 1.1, human operators and evaluators must be able to trust that the lethal autonomous weapons system (LAWS) under their charge will perform as intended to promote adoption. For machine learning (ML)-enabled LAWS, which is the primary focus of this guidebook, trust can be engendered by focusing on the principles of RAI (e.g., governability, traceability, reliability, etc.) and ethical conduct throughout the software development lifecycle (SDLC). Importantly, the trustworthiness and ethics of a system often depend on the development of a third feature: human-systems integration (HSI). This chapter covers these three topics to foreground how closely they are interrelated not only in theme, but in the life of products and in the testing, evaluation, verification, and validation (TEVV) process.1

2.1 Setting the Groundwork of Trust, Ethics, and HSI

Developers must design LAWS that can work alongside humans and for warfighters to use in an ethical manner as directed by humans. The LAWS, itself, does not decide on the ethical values that guide it. Machines, regardless of whether they are machine-learning enabled systems (MLES), fundamentally lack agency and cannot be held accountable for their actions. Rather, developers—aided by stakeholders—must decide on what ethical values the machine should follow before they begin designing the system. Good system design should enable all desired ethical values, and operators should be able to configure those values as appropriate before deployment and monitor them during operation.

Ultimately, therefore, developers and stakeholders must understand trust, ethics, and system design—in particular HSI—and the interrelationship between all three to design systems that generate user trust. In the following sections, we provide an overview of key terms and how developers and stakeholders must consider them to design and deploy systems with the highest levels of trust and trustworthiness.

2.1.1 Trust

Trust is inherently personal and highly dependent on human memories and perception; it is influenced by qualities and attributes that can vary over time and by situation. Because there are many ongoing studies on the nature and dimensions of trust, definitions of “trust” are numerous and varied. Because the purpose of CaTE is to develop the evaluation and measurement of trust in LAWS, we establish definitions for types of trust in the following sections that we further use in the guidebook.

The National Institute of Standards and Technology’s (NIST’s) website offers several useful definitions of trust, which share similar language. One definition that is useful to our discussions defines trust as “A belief that an entity meets certain expectations and therefore, can be relied upon.” Another defines trust as “The willingness to take actions expecting beneficial outcomes, based on assertions by other parties” [NIST 2005]. These definitions, while broad, provide a foundation for understanding and reasoning about how we approach trust.

Our companion guidebook *Human-Centric, Teaming-Focused Approach for Design and Development of Non-Deterministic Systems: A Human Machine Teaming Design Framework* provides an in-depth survey of definitions of trust and how we can use them in human-machine teaming (HMT) experiments [Rigsbee 2025]. That guidebook investigates nuances of different trust definitions and provides guidance for developing ways to measure trust in a broader way that is pertinent to HMT, which we apply to LAWS-specific experiments. We discuss these topics later in Section 2.3.

Other sources add the concept of "willingness to be vulnerable" to the definition of trust, which emphasizes the idea the that LAWS will be taking actions that have a direct impact on the user and that users need to have a willingness, consciously or subconsciously, about accepting those risks [Cox 2016]. This definition is significant because it changes the nature of evaluation questions from "Do you trust the robot to do X" to "Are you willing to accept risk Y when letting the robot do X."

The article "Trusting robots: a relational trust definition based on human intentionality," covers ideas of dependence, risk, positive expectation, and free choice, and it highlights that trust functions as an asymmetric relation [Schäfer 2024]. These dimensions highlight the complexity involved in constructing trust experiments and measuring trust. The asymmetric relationship means that robots do not have to trust humans, which emphasizes that trust models used in HMT are going to be different than those used between two humans. For that reason, proper development of these models requires the acquisition of new data, especially in operational context.

Different definitions of trust may be used in different situations to emphasize different dimensions of trust. In our companion guide, *Data Curation for Trustworthy AI*, we use the following definition: "A trustworthy AI-enabled system must be optimized for performance on the true distribution of inputs it will encounter in a deployed environment" [Clemens-Sewall 2025]. That definition emphasizes how a data distribution is fundamental to the meaning of trust, and we use it throughout that companion guide to provide coherence and focus for understanding trustworthy data. In this guidebook, we describe several approaches that developers can implement during development to positively influence the perception of trust. Some of these methods include developing feedback mechanisms, following expected behaviors, and running behavioral trials during design and development to gauge which features are most likely to engender trust. Section 2.3 provides an example of how we developed and applied a trust measure in a laboratory environment.

2.1.2 Calibrated Trust

When addressing the complex issues of ML, especially in LAWS, we feel that the concepts of trust that we began outlining in Section 2.1.1 are useful, but insufficient. To properly engage user trust in LAWS, it's important to develop a concept of trust that provides better information about users' expectations and reasoning and how trust can change depending on various factors. Better understanding of user trust enables developers and other stakeholders to make design decisions based on the situations and environments where LAWS operate, and it will promote the development of systems that better align with human expectations and behaviors.

For that reason, we want to introduce the concept of *calibrated trust* to drive discussions about how trust works with LAWS. The SEI offers the following definition of calibrated trust: "a psychological state of adjusted confidence that is aligned to end users' real-time perceptions of trustworthiness" [Gardner 2023]. To address trust, therefore, this definition guides us to gauge how trust fluctuates depending on a person's expectations based on past experiences and new situations.

In addition to addressing general circumstances under which trust takes form, this approach to understanding trust also addresses how trust changes under specific conditions, locations, and times. For example, to get a sense of general trust as opposed to calibrated trust, we may ask an operator the following question: "Based on the current feedback and behavior from the LAWS in these exact circumstances, do you trust it to make good decisions?" This question measures trust in a particular circumstance, but not how the user formed his or her thinking to form trust, or how that trust might change. Calibrated trust goes even further to understand how factors that contribute to the formation of trust can change dynamically with each interaction. Therefore, to gauge calibrated trust, we might ask an operator, "Based on similar situations you have experience with this LAWS, do you trust it to make good decisions in *this* circumstance?"

However, trust—because it is rooted in individual perception, mental states, and other conditions—poses significant challenges for objective measurement, both on an individual level and across populations. These challenges underscore the need for a clear framework to understand trust, particularly in environments involving complex technologies. Ultimately, the only way to be able to predict whether a person will trust a system is to run experiments involving trials with real users using the devices in operationally relevant situations.

2.1.3 Trustworthiness

While we can't easily measure trust, we can talk objectively about the *trustworthiness* of a system. The SEI offers the following definition of trustworthiness: "a property of a system that demonstrates that it will fulfill its promise by providing evidence that it is dependable in the context of use and end users have awareness of its capabilities during use" [Gardner 2023]. Such a definition alludes to the possibility for generating tangible evidence for developing testable methods to prove not only whether a system is trustworthy, but also just how trustworthy it is.

Based on this definition, and in its goal to promote trust, this guidebook helps its readers develop bodies of evidence at all stages of the SDLC to understand the trustworthiness of a system. Such a process would be akin to the way that TEVV personnel and others might build an assurance case. NIST defines an *assurance case* as "A reasoned, auditable artifact created that supports the contention that its top-level claim (or set of claims), is satisfied, including systematic argumentation and its underlying evidence and explicit assumptions that support the claim(s)" [NIST 2025]. The similarities are obvious, and we can see that an assurance case is a mechanism for supporting trustworthiness. In fact, the term *trust assurance* has been in use for many years to refer to the idea of building the trust case, usually in terms of cybersecurity. Throughout this document we pursue how to build a trust assurance case to support the trustworthiness of the LAWS for ethical deployment.

2.1.4 Ethics

Ethics is the set of principles that govern a person's actions. Like trust, ethical principles can sometimes be based on personal views and beliefs, and no system of ethics is universally held as valid. There are a variety of different ethical models such as consequentialism, deontology, and virtue-flaw, which we can use to help study and understand sets of ethics, how to relate them together, and how to reason about them. These models should not be mistaken for a universally accepted or established ethics. IEEE 7000-2021 offers a good discussion in one of its appendices of ethical models in the context of software systems [IEEE 7000-2021]. That document describes how to apply, create, and design processes that embody and reflect ethics.

Like IEEE 7000-2021, this guidebook does not describe a particular set of ethics. Rather, it explains how development teams might reason about ethical models to incorporate them into development processes and system designs. Knowing how development teams consider ethics during design and how TEVV personnel evaluate them during testing is central to the approval process in DODD 3000.09. Therefore, this subject is of particular interest to TEVV personnel, and it will play a central role in building the trust assurance case.

2.1.5 Human-Systems Integration (HSI) and Human-Machine Teaming (HMT)

Human-systems integration (HSI) and human-machine teaming (HMT) are areas of knowledge that pertain to how an operator or warfighter will interact with these systems and, ultimately, to whether they will trust or distrust them. The central focus of CaTE is operator trust and HSI and HMT are therefore core concerns. Our companion guide *Human-Centric, Teaming-Focused Approach for Design and Development of Non-Deterministic Systems: A Human Machine Teaming Design Framework* and *Human Systems Integration Test and Evaluation of Artificial Intelligence Enabled Capabilities: What to Consider in a Test & Evaluation Strategy* provide detailed background on definitions and descriptions of HSI and HMT [Rigsbee 2025; CDAO 2024b]. They are consistent in their message that development teams must consider operators and warfighters throughout the entire product development lifecycle, and testing processes must also take them into account.

2.2 Trust, Ethics, and HSI Across the Product Lifecycle

In the following sections, we build on the definitions of trust that we offered above by discussing how issues related to ethics might impact trust in LAWS. Throughout the guidebook, we provide ways of addressing these problems whenever TEVV personnel might encounter them, and we link references to the sections in the guidebook where they can find those solutions.

It is important to note that trust, ethics, and HSI are highly interrelated and crosscut many development and testing activities. Therefore, addressing any issues related to trust, ethics, or the development of HSI requires holistic approaches that encompass the entire life of the product. Developing such approaches is difficult because systems take time to develop, and cultures and processes take time to evolve. In addition, it is highly likely that the systems that developers and TEVV personnel are developing and testing today may not have followed many of the RAI or ethical practice guidelines that are currently available. Almost by definition, very few systems are developed using the “latest” practices.

However, it isn’t too late to take advantage of these practices and processes, even if retroactively applied, and they can be used to highlight and prioritize where to spend evaluation effort. In this guidebook, we inform TEVV personnel about those latest practices with the intent that they can begin making use of them today. However, we understand the level of effort that implementing such practices might entail. Process adoption is difficult, and it requires significant time and effort. While ethics as a philosophical reflection is often a high-level aspiration, ethics expressed in detailed engineering processes, especially software and AI, is a recent investment, and the processes and methods to adequately test it require extra attention from TEVV personnel.

2.2.1 Ethical Requirements May Not Be Explicit

LoW and laws of armed conflict (LoAC) provide warfighters with ethical guidance, but technical systems need clearly defined system requirements [OGC 2023; NSLD 2022]. However, as of the writing of this document, no guides on how to write requirements for ML-enabled LAWS and their targeting systems are available, even though efforts are underway to bridge the requirements-engineering gap for autonomy [Turri 2022]. In Chapter 3, we address approaches for defining system context, what issues to look for, and how to understand hidden requirements.

In Chapter 4, we discuss training and validation data and how it must address ethical situations that the ML components must deal with. For example, if a set of ethical values focuses only on static visual characteristics of objects and entities (e.g., a badge), then developers do not need to design motion-tracking models with these properties in mind. However, if the system makes judgments about an object’s behavior and actions over time to make ethical decisions, then the system must have a way to detect and track an object’s behavior. System designers and developers must reflect these ethical requirements in system capabilities, and it is up to the TEVV staff to confirm that the requirements are complete. If they are incomplete, then the TEVV personnel must take appropriate steps to supplement and amend the missing requirements.

2.2.2 Data Is Central to Ethics and Trust

The quality of any learning system can only be as good as the quality of the data that developers use to train it. Developers must carefully consider how to collect and curate the data to meet not only the detailed mission requirements, but also its overall ethical needs. We provide details on this topic and on how developers can collect and curate training data for these special needs in Chapter 4.

The RAI principle of equitability refers explicitly to data in terms of bias. However, when it comes to ML-enabled LAWS, bias becomes even more important with regards to ethical decision-making because the consequence of bias can be more severe.

Risk and data are highly intertwined in learning systems, so it isn’t surprising that artifacts such as the Center for Naval Analyses (CNA’s) dimensions of autonomous decision-making (DADs) play a central role in determining dataset quality [Stumborg 2021]. We recommend that developers and TEVV personnel use DADs during testing and requirements gathering because answering the questions in the DAD document significantly improves these processes. For example, the fifth DAD, “Command and Control,” asks the following question: “Is the IAS [intelligent autonomous system] prohibited from initiating operation in the absence of a control link to a human operator?” Understanding the relationship between human operators and autonomous behavior and when control is lost—as this question prompts TEVV personnel to do—significantly improves the trustworthiness of the system. We provide more in-depth discussion of DADs and how TEVV personnel can make use of them in Chapter 3.

2.2.3 Methods for Ethical Measurements May Be Inconsistent, If They Exist at All

On Oct 16, 2024, DARPA kicked off a new program called “ASIMOV: Autonomy Standards and Ideals with Military Operational Values.” The purpose of the program was “to develop benchmarks to objectively and quantitatively measure the ethical difficulty of future autonomy use cases” [DARPA 2025]. DARPA is known for tackling complex and hard-to-solve problems, and the fact that they have decided to initiate a multi-year program in this area emphasizes the difficulty of measuring ethics. Given that the program won’t reach completion until 2026, quantitative measures are unlikely to be available for some time.

As of the writing of this document, there are no clear quantitative measures and metrics for judging ethical behavior. Without clear measures, metrics, and thresholds that can translate into requirements in a computer-readable form, testing whether LAWS behaves ethically is extremely challenging, and lack of such measures will likely lead to inconsistencies with different providers using different approaches that will yield different results. Most efforts for the novel development of autonomy in LAWS employ ML, which uses model-level measures such as precision, accuracy, and reliability. However, even though these measures are useful at the internal model level, they are insufficient at the system level.

In chapter 3, we discuss methods for writing system requirements, which is especially important—and difficult—when such requirements with respect to ethical behavior are missing. Our discussion provides guidance to clarify how TEVV personnel can generate requirements for testing these difficult aspects of the system. Further, in chapter 5, we discuss principles and approaches for increasing traceability in the system, how the system shows details about how it makes decisions, and how to observe and control various measures and metrics.

2.2.4 Process May Not Align with Goals

Development and testing processes may not have kept up with changing business goals and emerging standard practices, and TEVV personnel should not assume that existing processes will be sufficient for ML-enabled LAWS. Before beginning new development efforts, we recommend that development and TEVV teams perform a detailed evaluation of all their development and testing processes in light of the most recent guidance and that they update their development and testing processes as time and resources allow [DOD 2022a; Dunnmon 2021; IEEE 7000-2021; IEEE 7001-2021; ISO/IEC 5338:2023; Stumborg 2021].

Process adoption, cultural change, and getting funding for engineering are always a challenge, and we expect that these tasks will prevent developers from adopting all, or even many, of the available guidance documents on early ML-enabled LAWS. Therefore, we recommend that development teams prioritize all process changes based on the specific system's needs and mission risk. In chapter 3,we discuss how TEVV personnel can map project goals into requirements. In Chapter 8 and Chapter 9, we discuss developmental testing and evaluation (DT&E) and operational testing and evaluation (OT&E), respectively.

2.2.5 You May Need to Update Test Plans to Deal with Ethics

Unsurprisingly, there are no documents that offer clear instruction for how to specifically test for ethics. The Director of Operational Test and Evaluation (DOT&E) has produced guidance documents for test plan development and for testing AI, and that guidance requires that development teams address ethics and RAI. However, it lacks specific criteria by which to do so. Further, these guidance documents are limited in their scope to OT&E and live fire test and evaluation (LFT&E) [DODM 5000.100; DODM 5000.101]. We expect that other organizations will develop further guidance to address developmental testing, evaluation, and assessments (DTE&A).

For now, the traditional practice of developing test scenarios around corresponding requirements is the best method available for testing ethical requirements. If explicit ethical requirements are not provided at the outset of testing, then TEVV personnel will need to backfill them during design and prepare tests appropriately. We offer several discussions in this guidebook to support such work, including the following:

- Chapter 3 discusses techniques for developing ethical requirements.
- Chapter 5 provides guidance on instrumentation that may be available for evaluation.
- Chapter 6 discusses how to test the individual models.
- Chapter 8 describes how developmental testing might provide useful insight before OT&E.
- Chapter 9 covers the current practice for OT&E.

2.3 Measuring Trust: Key Elements of a Trustworthy System

As we discussed earlier, trust, as a general concept, is subjective and difficult to measure objectively. Therefore, at the inception of the CaTE pilot, we had no clear way to measure trust in a contextualized way. This issue made it difficult to make progress because, as outlined in the CaTE background, our objective was to "ensure high trust and assurance" for ML systems. Our initial effort, therefore, involved developing a basic metric to measure the trust that users placed in a system.

These early efforts of the CaTE pilot resulted in a study that we published in March, 2025, titled, "Center for Calibrated Trust Measurement and Evaluation (CaTE) Pilot: Human-Machine Teaming

Study 1" [Hale 2025]. That study showed how it is possible to measure whether a set of software or hardware users is more likely to trust one approach to system design and behavior over another. For example, we can test two different interfaces to determine—with a significant degree of confidence—whether users are more likely to trust one over the other.

In the following sections, we provide an overview of that study's findings. We discuss the software features that we determined are most important for generating user trust. The purpose of sharing these results is to give development teams a better sense of what features they should prioritize for LAWS to promote trust, and to provide a better sense of how users might think about system interaction so that development teams can better plan how to build the system.

2.3.1 System Trustability Scale: How to Measure Trust

As mentioned above, one of the first steps we took was to develop a metric that we could use to measure users' trust in a system. To do so, we leveraged John Brooke's System Usability Scale (SUS), which he developed to "take a quick measurement of how people perceived the usability of computer systems" [Brooke 2013]. Using SUS as a starting point, we developed the System Trustability Scale (STS) to measure user trust in systems. We then applied STS to measure trust for a hypothetical system and to determine whether STS worked correctly to serve its intended purpose.

STS works by engaging users to evaluate a series of 10 statements, which are as follows:

- I think that I would like to use this system frequently.
- I found the system unnecessarily complex.
- I thought the system was easy to use.
- I think that I would need the support of a technical person to be able to use this system.
- I found the various functions in this system were well integrated.
- I thought there was too much inconsistency in this system.
- I would imagine that most people would learn to use this system very quickly.
- I found the system very cumbersome to use.
- I felt very confident using the system.
- I needed to learn a lot of things before I could get going with this system.

Our study presented these ten statements to SEI employees. Then, we asked them to imagine fictitious scenarios involving different kinds of systems. In one scenario, we asked study participants to think through their possible engagement with a fully autonomous, lethal quadruped robot.

We have taken note of the fact that testing these systems with SEI employees as potential users is not the same as testing them with actual users of the system. However, SEI employees are well acquainted with DoD missions and objectives, and their input is significant as an initial effort to assess trust and to develop further studies. We intend to continue studying user trust with different pools of participants, but these initial findings are useful as a preliminary way of thinking through system features and behavior, and their relationship to user trust.

2.3.2 Key Findings from the SEI's Trust Study

The study identified 12 areas that engender user trust, which we reproduce in Table 1. The table lists the features in order from most important for engendering trust—as identified by users—to least important. The table also includes an explanation of the feature and requirements for implementing it in the system. We offer a discussion of these findings in Section 2.3.3.

Table 1: Summary of Key Findings from the SEI's Trust Study

Priority	Feature	User Needs and Trust Requirements
1	Human authorization of force	• **User need**: Users expect human authorization in all decision-making processes leading to lethal force. • **Trust requirement**: Development teams must implement human-in-the-loop for all decision-making involving the use of lethal force.
2	High-precision targeting	• **User need**: When AI is involved in targeting, users expect it to meet or exceed human capabilities. • **Trust requirements**: The system should perform at 98-100% accuracy.
3	Testing and reliability	• **User need**: Users value evidence over claims of testing and reliability. • **Trust requirement**: TEVV personnel must provide proof of successful and thorough testing, such as thousands to tens of thousands of trials in varied environments and scenarios to demonstrate reliable performance in harsh conditions.
4	Situational awareness	• **User need**: Users want clear and actionable sensory information and data to make mission critical decisions when monitoring, cooperating with, or controlling a system. • **Trust requirement**: Systems must provide operators with real-time, easy-to-understand data (e.g., video, audio, system location, orientation, projected path, objects of interest, obstacles, battery status, etc.).
5	Hardware redundancies	• **User need**: Users expect systems to have redundancies and fail-safes to prevent failure. • **Trust requirement**: Systems must include redundancies for critical components, especially for navigation and classification (e.g., camera, mic, speaker, GPS, LiDAR, or IMU) and features that enable a total system shutdown (e.g., a remote or on-system, analog kill switch).
6	Operator handover in uncertain contexts	• **User need**: Users want to be able to intervene and make critical decisions when confidence is low or uncertainty is high. • **Trust requirement**: Development teams must implement clear notifications, error messaging, and handover protocols to decrease the likelihood that the system will make poor decisions and cause unintended harm.
7	UI to support operator decision-making	• **User need**: Users want an intuitive UI that creates visual separation between targets and background, coupled with precise, easy-to-parse data that reduces cognitive load and improves performance. • **Trust requirement**: The system must employ bounding boxes for targets and provide simple but precise class labels and confidence metrics.
8	Ethical governor	• **User need**: Users want transparency about system limitations and motivations. Although skeptical about its implementation and effectiveness, users view an ethical governor as a system requirement.

		• **Trust requirement**: Development teams must design, implement, and test an ethical decision-making framework that clearly explains system limitations to users.
9	De-escalation capabilities	• **User need**: Users want systems that prioritize de-escalation and that minimize the use of lethal force. • **Trust requirement**: The system must prioritize de-escalation via one of the following: (a) communications; (b) non-lethal engagement; (c) precision targeting to wound rather than kill. The system applies lethal force only when absolutely necessary and in appropriate environments.
10	Spatial, rules-based framework for autonomous behavior	• **User need**: Users do not believe current AI and ML can safely or reliably classify human targets based on appearance or behavior. Users feel comfortable mapping target classification risks onto environments but not lives. Users expect a system to adhere to predefined rules that determine what actions are allowed within different operational environments. • **Trust requirement**: Development teams must implement a spatial framework for users to define operational zones that determine levels of autonomy and restrict robotic behaviors ensuring reliable, context-aware behavior. We recommend the following zone classifications: combat, liminal, non-combat, and no-go.
11	Visual and audio operator feedback	• **User need**: Users want real-time, high-quality feedback to understand the robots' location and actions. • **Trust requirement**: Development teams must equip the system with low-latency, high-fidelity cameras, microphones, and speakers to increase users' connection with and confidence in the system.
12	Post-mission review and continuous feedback	• **User need**: Users expect an intelligent system to learn from past missions and to improve over time. • **Trust requirement**: Development teams must design the system to log mission data and conduct-rigorous, post-mission reviews to improve model performance and identify or mitigate errors and unintended biases, ensuring that models stay current and aligned to evolving operational needs.

2.3.3 Discussion of the SEI's Trust Study

The following sections group the features listed in Table 1 into categories based on whether the feature impacts human operators, the system, or the system framework. We then offer a discussion about each of these areas and their impact on ML systems and LAWS.

2.3.3.1 Human: Oversight and Decision Support

The features we tested in this study that directly affect humans operating the system are as follows:

- human authorization of force (priority 1)
- operator handover in uncertain contexts (priority 6)
- UI to support operator decision-making (priority 7)
- visual and audio operator feedback (priority 11)

Unsurprisingly, for a study on building trust in human-autonomous systems, the feature that ranked as highest priority focuses on human oversight and visibility into decision-making and operator control of that decision-making. At their core, autonomous devices use AI components (and, increasingly, they use ML components as well) to make decisions, and operators want to know why they make those decisions, and they want to be able to influence them. This concept reinforces the RAI principles of traceability and governability in the development of LAWS.

Importantly, quality human oversight cannot be “tested in” at the end of the development process. Rather, it must be designed in to be effective. We cover system context and requirements definition in Chapter 3. However, TEVV personnel can consult IEEE 7001-2021 “IEEE Standard for Transparency of Autonomous Systems” to review an example of modern approaches to promoting observability [IEEE 7001-2021]. In Chapter 5, we discuss how ML impacts system design.

2.3.3.2 System: Reliability, Robustness, and Continuous Improvement

The group of features that impact the system properties such as robustness, reliability and system safety properties are as follows:

- high-precision targeting (priority 2)
- testing and reliability (priority 3)
- hardware redundancies (priority 5)
- post-mission review and continuous improvement (priority 12)

Three of these four features manifest throughout the entire system development lifecycle because development teams must design these features into the product and into the development process to successfully implement them. Consequently, most chapters in this guidebook address testing and reliability in some fashion, whether it be through requirements, design, development, or any of the stages of testing.

High-precision targeting is an example of a specific place to test for reliability. The study indicated that the system should meet or exceed human capabilities for it to engender trust, and the measure of that requirement was for the system to achieve 98-100% accuracy, which involves both hardware precision as well as targeting-system precision. The perception, targeting, and response (PTR) chain—which we describe in depth in Chapter 5—is responsible for processing the input sensor information and resolving it into targeting identification and final response actions.

The study also claims that “Falling short of accuracy expectations for target identification, classification, and engagement will prevent or erode trust in the system” [Hale 2025]. ML components are primarily responsible for this series of tasks, and, therefore, targeting precision should be one of the critical metrics for TEVV.

The “testing and reliability” feature is also of particular interest because respondents feel that, to earn trustworthiness, the device must have been extensively tested, which reinforces the idea that we are building a trust case. In conversations we held during the study, operators expressed the need to train extensively with these systems before they could trust them, which is not surprising since extensive training is a common approach for building trust.

2.3.3.3 Framework: Ethical and Autonomous Behavior

The final category lists features that impact the framework of the system; these are as follows:

- situational awareness (priority 4)
- ethical governor (priority 8)
- de-escalation capabilities (priority 9)
- spatial, rules-based framework for autonomous behavior (priority 10)

This category covers general autonomous behavior including targeting and movement. This behavior forms part of a device's ethical conduct. General perception is still a central part of these behaviors, but localization is also an important consideration. If the devices have spatial-based rules, such as no-go zones, then it is important to devise a reliable and robust way to identify these areas. GPS has downfalls because attackers can mislead it, and attackers can use visual camouflage to mislabel buildings or areas.

In addition, development teams must provide a way to deliver ethical conduct holistically through the entirety of the design, production, and implementation of the system to provide the needed level of robustness. In Chapter 3, we provide guidance on how to describe ethical requirements. In addition, development teams can consult IEEE 7000, "IEEE Standard Model Process for Addressing Ethical Concerns during System Design," for guidance on how to perform ethical development [IEEE 7000-2021]. In Chapter 5, we provide an overview for encoding and expressing ethics in a system design.

3 System Context and Requirements

This chapter provides guidance for defining systems context and producing good requirements for machine learning (ML) and ML-enabled systems (MLES), especially to support the process of testing, evaluating, validating, and verifying (TEVV) such systems. It is common knowledge that the TEVV of systems is much improved when system context descriptions and requirements are well-documented, and that relationship holds for MLES as well. When requirements are not well-documented, it is often up to evaluators to backfill them to the best of their ability and then communicate the extent to which their results support the intended case. We understand that it is rare for the development process to provide clear, concise, and unambiguous requirements that testers can directly use to test the system, especially with the introduction of radically new technologies, such as artificial intelligence (AI) and ML. In light of these difficulties, this chapter offers guidance on determining if requirements are complete and provides steps to take when they are not. Finally, this chapter addresses how to practically move forward when missing critical information.

There is a significant number of sources that offer guidance for documenting system requirements. However, most of these sources don't go far beyond high-level guidance, often resorting to impractical generalizations and abstractions. This deficiency is not surprising since the nature of guidance is to be broad, and the science, technology, techniques, and engineering discipline are still evolving. As a result, we have not found a substantial corpus of examples or guides for describing mission statements regarding the decision-making components of MLES, which is a substantial need for ML-enabled lethal autonomous weapons systems (LAWS). Therefore, the challenge for TEVV, in the interim, is to extend the broad guidance that exists into more detailed, actionable information and into real metrics. There are a variety of robust structured analytic methods, underused by the autonomy community, that can be used to close the gap. This chapter builds on those methods and offers guidance to produce necessary information and metrics to support TEVV.

3.1 Recommendations for System Context and Requirements

This section provides recommendations for personnel to perform TEVV of LAWS. Supporting information and further details can be found in Section 3.3.

3.1.1 Use the Defense Innovation Unit Worksheets When Defining Mission Use

The Defense Innovation Unit (DIU) guidelines and worksheets, detailed below in Section 3.3.1, provide useful information for any project. Many of the questions they provide pertain to how the system will be developed, but they can be used retroactively to see how it was developed. Using them in this way provides needed inputs for test development. Many current guidelines ask useful questions, but the commentary sections in these worksheets provide further explanations about the questions and how to provide a basic evaluation of the quality of the answers.

3.1.2 Use Checklists Such as Dimensions of Autonomous Decision Making (DADs) to Identify and Elicit undocumented TEVV Needs

TEVV personnel should maintain a good list of test-specific questions to use for evaluating MLES. As we discussed in Section 2.2.2, the document “Dimensions of Autonomous Decision Making” describes 13 different types of questions (which, as the title of the document suggests, are referred to as dimensions of autonomous decision-making (DADs)) with a total of 585 specific questions that TEVV personnel can ask about the decision-making process developers used to develop the device. A robust domain-specific set of questions can then be used to evaluate test coverage and completeness. See Section 3.3.11 for a further discussion of DADs and Section 3.3.6 for a process to support the use of questions like these to develop test indicators and metrics.

3.1.3 Choose and Consolidate Analytic Methods to Identify Appropriate Metrics

TEVV personnel can use analytic methods (see examples in Section 3.3 below) to develop appropriate test metrics from abstract requirements. Organizations should choose one method that fits their needs and apply it to all requirements to provide consistency across TEVV activities. Consolidating all processes into one method reduces training time and costs and improves communication between personnel because they can share techniques and terminology.

Value Criterion Indicator Observable (VCIO) is an iterative method for turning abstract socio-technical characteristics into observables. We discuss VCIO in Section 3.3.12. In addition, we discuss the Goal Question Metric (GQM) and the Goal Question Indicator Metric (GQIM) in Section 3.3.6 with an example of how to map the “Civilian Harm Mitigation and Response” goal to ML-model metrics. Systems Theoretic Process Analysis (STPA) and Failure Mode and Effects Analysis (FMEA) are commonly used safety practices for developing a safety assurance case and could also prove useful for TEVV personnel.

3.1.4 Check If the System Safety Analysis Factors in the Function of ML Faults

When performing a system safety analysis and working to understand how the system’s faults function, it is important to note that ML algorithms have their own definitions and metrics for faults. TEVV personnel will need to incorporate these in the safety analysis to ensure as many functions as possible are covered by these measures. We provide expanded definitions in our discussion of the system safety assessment (SSA) in Section 3.3.5. As of this writing, many safety standards such as MIL-STD-882E are under revision, and they do not yet include these definitions. See Section 3.3.5 for more details on fault detection.

3.1.5 Integrate Context-Specific Emergency Stop Mechanisms

Introducing ML algorithms into the design and operation of LAWS also introduces new variability to decision pipelines dependent on these algorithms. It is not feasible to predict how an ML algorithm will respond to circumstances it was not specifically trained to understand. Such circumstances may cause the algorithms to provide incorrect information to the system, resulting in misinformed decisions that can potentially lead to irreversible harm.

To help mitigate consequences from these situations, requirements for LAWS should include emergency stop mechanisms, such as deactivation switches and manual overrides. Developers and TEVV personnel should use the LAWS' deployment context to determine the amount, type, and locations of these mechanisms. Requirements for emergency stop mechanisms should also consider all ML components involved in decision-making pipelines.

3.1.6 Leverage Post-Event and Live-Logging Mechanisms to Provide Traceability for ML-Dependent Processes

Because traceability is a core area of the DoD's Ethical Principles for AI, requirements for LAWS should include mechanisms for providing such traceability, specifically for processes that leverage ML algorithms. There are two ways developers can provide such traceability: by recording logging decisions for post-operation analysis and by surfacing decision information in real time during system operation. Requirements for LAWS, therefore, should include mechanisms to capture and avail decision-making data in both ways, and they should also specify a number of data points and decisions that the system must log or surface to operators during system use at a frequency and detail that helps them fully understand how those decisions were made. These requirements should target the highest impact decision pipelines and focus on their core components. As we noted in Section 3.3.10, developers should consider operator cognitive load when determining what information to surface to the operator during system use. Developers should also consider how the system will be used, who will be using it, and how decision information will be delivered to operators when writing these requirements.

We offer further discussion about this recommendation in Section 3.3.14.

3.1.7 Test the System in All Conditions Outlined by Performance Requirements

TEVV personnel should develop a test for every requirement that development teams or others document for the ML system. For example, if there is a requirement about the daytime performance of an object-detection model, there should also be a repeatable test with a specific dataset that demonstrates the system meets the requirement. The same approach should apply for nighttime or rainy-day performance requirements.

TEVV personnel should consider results across all conditions individually and not average them together because success in one set of testing conditions is not evidence for success in another set of testing conditions, regardless of the similarity between conditions. Similarly, every use case should have a test and test suite. If the system uses two operating domains with different safety constraints, each should have its own test suite that matches the requirements for operating in that domain.

3.2 Observations About Model Metrics

When testing systems, TEVV personnel might be inclined to test the individual models for metrics such as accuracy, precision, recall, mean average precision, and so on. While this is appropriate for standalone model development, the models in LAWS are operating within a larger system context that testers must consider. For example, testing whether a single object detector has a high chance of identifying an adversary does not imply that the remainder of the targeting and response chain performs well.

3.3 Commentary on System Context and Requirements

This section provides in-depth explanations for the recommendations and observations listed above.

3.3.1 DIU Responsible AI Guidelines in Practice

The DIU's "Responsible AI Guidelines in Practice" and its accompanying worksheets provide an excellent introduction, overview, set of guidelines, and commentary for applying the Department of Defense's (DoD's) responsible AI (RAI) principles for developing a product [Dunnmon 2021]. These artifacts tend to be general because they are intended for anyone working on AI in the DoD for any purpose, so they do not contain specific details regarding LAWS. However, the questions posed in these guidelines and worksheets are relevant, so providers should not only answer them, but offer evidence that they followed the worksheets to contribute to their assurance case and body of evidence for certification.

Each worksheet contains questions to help perform AI development planning and tasks, and these questions are accompanied by a section with commentary that provides useful guidance to a reviewer. For example, the Planning worksheet asks, "Have you clearly defined tasks, quantitative performance metrics, and a baseline against which to evaluate system performance?" The commentary explores the terms *clearly defined, quantified metrics*, and *baseline* along with some additional questions. Again, due to the breadth of possible AI systems that the guidelines are intended to address, they do not extend to some of the details needed for a LAWS. For example, the commentary about "clearly defined tasks for AI systems" addresses how to determine if an AI approach is needed but does not specify thresholds for LAWS performance [Dunnmon 2021].

As mentioned before, due to the current sparsity of ML-enabled LAWS deployment, detailed guidance is not currently available, but we hope to see an extension to the DIU Guidelines —or others—that specifically target ML-enabled LAWS.

3.3.2 ML Throughout the System

Because so many components can use ML, there are many approaches to how and where systems can make use of it. These complexities change how we design, develop, and test the system. Currently, during perception and targeting, it is reasonable to use different strategies or techniques for object identification, classification, object tracking, scene understanding, target nomination, and response selection. To provide a rough analog, a system might employ a separate piece of ML for each step of the find, fix, track, target, execute, and assess (F2T2EA) chain [JP 3-60]. In addition, TEVV personnel should consider what models the navigation system employs, such as goal planning, waypoint determination, or trajectory planning because these different pieces require different detailed requirements.

ML models make predictions about their inputs, such as if an entity should be classified as a tank, but do not make decisions themselves. The surrounding program logic compares these predictions against thresholds and then directs other parts of the system to take some action. When testing, TEVV personnel must consider not only the individual components, but also the surrounding symbolic logic and ultimately the behavior of the entire system. Therefore, while it is informative to examine the output of a first-pass object detector for evaluation and diagnostic purposes, the final TEVV effort should examine the risk associated with the question of whether the system took action against the targeted entity and whether that action was appropriate. The final assurance case for ML-enabled systems will be extensive when all the outputs are considered.

The document *Reference Architecture for Assuring Ethical Conduct in LAWS* describes an example of a system that one can use to consider and reason about a sufficiently complex autonomous system that identifies both traditional navigation pieces and an advanced, fully autonomous perception, targeting, and response (PTR) chain [Mellinger 2025]. It goes into detail on how the components can be organized to promote testability, usability, and observability (among other properties) and introduces patterns and components that promote ethical conduct governance for trustworthiness. Figure 2 shows ML components (the ones in green) that developers and designers can potentially use for architecting a PTR chain.

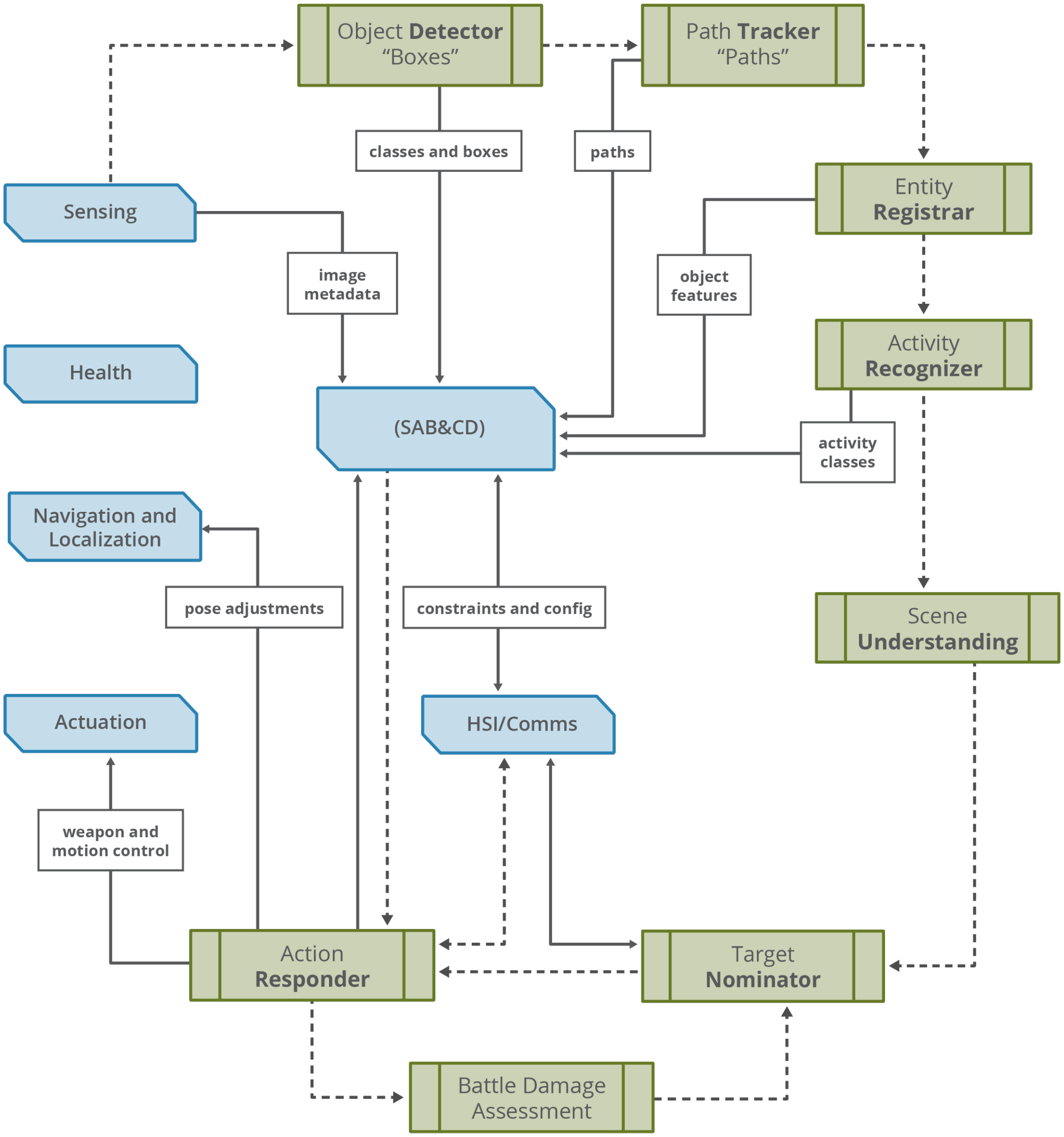

Figure 2: Example of a Perception, Targeting, and Response Chain

3.3.3 Cognitive Dissonance in the PTR Chain

For the time being, we assume that PTR consists of multiple components, learned or symbolic, that perform different actions such as basic object detection in a frame, object tracking across frames, object-to-entity mapping, scene understanding, target nomination, and response determination. Each component fulfills a different role in this structure, and each may have significantly different models, approaches, validators, and checkers. In fact, we expect that the system will employ multiple algorithms or approaches to increase reliability and robustness even within a single step in the chain. This complex chain of components results in a rich sequence of data that generates many predictions about that data (e.g., class) and assessments about the quality or validity of the data that contribute to a complex decision-making component. A deeper description of the types of components, their responsibilities, and recommendations for how to understand and test them can be found in the commentary in Section 3.3.2 of this chapter, Chapter 5 of this guidebook, and in the *Reference Architecture for Assuring Ethical Conduct in LAWS* [Mellinger 2025].

In later stages of the chain—such as during scene understanding, target nomination, and response—algorithms assemble many of these different predictions and assessments together to produce a result such as identifying an object as a hostile target or protected entity. During these steps, some of the indicators may be contradictory. For example, the basic object detector may provide erroneous classifications based on individual frames. The fact that some of the still-frame classifications are not accurate comes to light when the tracker evaluates them in a sequence to determine a path. In such a case, performing sensor validation could help determine which of the classifications in the sequence the system can appropriately discount and when.

Another example might occur during "scene understanding," in which the perception system analyzes the relationships between objects and other cues from the scene such as background information. During that process, the system might identify a quadruped animal as either a cat or a horse. When located in a scene beside a human, the "scene understanding" component can use the relative sizes between the two figures to choose the cat or horse label with greater accuracy.

To provide yet another example, adversarial ML (AML) or counter AI—where adversarial actors use mathematical techniques to modify training data or to generate data, such as images, that can be physically produced as objects such as glasses, t-shirts or, stickers for placement in the physical environment—can cause the AI to misclassify the objects [VanHoudnos 2024; Sharif 2016]. In these cases, researchers have found that some techniques only fool a subset of the classifiers, and a generalized attack is not yet available. To offer robustness against such attacks, developers can use other types of classifiers, symbolic or ML, to perform additional redundant classifications for verification.

In these complex systems, all these examples serve to illustrate how certain issues can result in a sort of AI "cognitive dissonance," where algorithms disagree on what should be resolved. In a complex sequence like the PTR chain, system designers should reflect on and consider any conflicting information between components that can result in such "cognitive dissonance." System designers must determine satisfactory confidence scores for discrepancies and how many discrepancies are acceptable. They must also determine whether these discrepancies lean toward particular biased outcomes. For example, when classifying civilians in certain situations, we are likely to bias toward a civilian classification rather than a hostile one, However, other situations might dictate that we should bias toward hostile classifications. Providing the operator with a clear set of thresholds and continuous feedback on the system's performance—relative to those thresholds—will significantly improve the operator's understanding of the machine state and engender trust.

Ultimately, system designers must be able to answer the following questions:

- What is the result of "excessive" discrepancies?

- Are the discrepancies the result of sensor failure or algorithm failure?
- Could the presence of many discrepancies be the result of an adversarial attack?
- How is the presence of many discrepancies communicated to the operator and maintenance individuals? And at what point does the system conclude that it is operating outside of its specifications?

Advanced systems will employ decision-making functions that consider all input values instead of relying on just one component. In these systems, combinatorial testing approaches are important because of all the permutations of the different factors that comprise a PTR chain [Chandrasekaran 2023].

3.3.3.1 Evaluation Criteria

The following list provides a set of considerations to address cognitive dissonance:

- The system should explain how PTR decisions are made, and how it deals with conflicting indicators. It is preferable to employ a mechanism that considers all attributes holistically rather than just choosing a single best value, as described in UL 4600 Section 8.3.3 [UL 4600].
- It is preferable to employ a combinatorial testing approach that exercises a full set of expected and prioritized mission scenarios. See Section 3.3.7 for examples of different environments.
- Designers should include defenses against adversarial attacks and counter AI in requirements, data sets, and test plans.
- Designers should consider how the system uses, or should use, redundant checks, voting schemes, or other hybrid ML and symbolic checks for defending against counter AI. They should also find ways for conflicts to be resolved in alignment with mission biases.

3.3.4 Incorporating and Evaluating Fail-Safe Designs

Fail-safe modes are common features of a safe design. Any traditional weapon system is expected to employ a variety of fail-safe mechanisms. However, developers and TEVV personnel that want to introduce ML into a system might need to reevaluate traditional fail-safe mechanisms before using them in such a system because those fail-safe designs may have assumed a human operator and their needs and behaviors. Such designs, therefore, might not align with the requirements and function of ML. For example, if a weapon intended for human use is mounted on an autonomous system, designers must consider how preexisting weapon-safety measures, assumptions, and human training and reasoning integrate into the targeting and firing process of the automated design for the weapon.

In addition to mechanical techniques, designers must also consider process-based, fail-safe designs. Such designs might draw on the idea introduced above about how the behavior of autonomous systems should align with human training and reasoning on weapon-safety measures. For example, it would be useful for a robotic weapon to point to the ground when it isn’t actively ready to engage. This measure is called “holstering,” and it mimics one of the four primary rules for gun safety that human users are familiar with [NSSF 2024]. Not only does this measure provide a fail-safe mechanism, but it also increases trust by visually communicating readiness and intent to the operator and other persons.

Autonomous systems might also employ other common fail-safe measures such as non-visible design patterns or redundant systems. All ML-enabled systems should have non-ML validators, checkers, and constrainers as part of early detection and safety checks. See the *Reference Architecture for Assuring Ethical Conduct for LAWS* for an in-depth example of how to build these measures [Mellinger 2025]. Some ML systems employ redundancy patterns by using multiple versions of models that are trained with different neural net architectures, subsets of data, or training practices with the results combined to form a consensus for ML redundancy. If non-ML systems are available, they can also be run in parallel. See Chapter 5 and the *Reference Architecture for Assuring Ethical Conduct for LAWS* for more details.

3.3.5 Fault Detection

Fault detection is imperative for any safety or mission-critical system and is foundational for ensuring that systems are robust, resilient, and, if necessary, degrade or fail gracefully. To frame this discussion, we differentiate faults[2] and failures[3] using the definitions in SAE's "Guidelines for Conducting the Safety Assessment Process on Civil Aircraft, Systems, and Equipment" [SAE ARP4761a]. Faults and failures are related in that an item or system can experience faults yet not fail (i.e., undergo a loss of function). The more faults or anomalies an item or system can withstand while continuing to operate within its specifications, the more robust and resilient that item or system is. This resilience is referred to as fault tolerance.

It stands to reason that a system's fault tolerance is a byproduct of the system's architecture implementation—both hardware and software—and its ability to detect and handle faults. It is, therefore, important to perform SSA activities early in the system concept- and architecture-development phase. A substantial amount of literature exists that can provide guidelines, methods, and processes for developing fault-tolerant systems. For example, SAE's Aerospace Recommended Practices (ARP) provide safety and certification consideration for highly integrated or complex systems [SAE ARP4754b; SAEARP4761a]. These ARPs provide guidance on SSA activities such as performing fault tree analysis (FTA), fault hazard assessment (FHA), failure mode effects analysis (FMEA), and common cause analysis (CCA), among others. While these ARPs are geared toward aircraft, the practices and considerations are applicable to the development and verification and validation (V&V) of any software-intensive system. It is worth noting that these standards are developed by SAE International: The Engineering Society for Advancing Mobility Land Sea Air and Space, and they are therefore relevant domains for the DoD. While the ARPs do not preclude development of ML-enabled systems, they also do not provide specific guidance on their development or V&V. Development and V&V of ML will be the focus of the remainder of this section.

[2] SAE defines the term "fault" as an undesired anomaly in an item or system [SAE ARP4761a].

[3] SAE defines the term "failure" as a loss of function or a malfunction of a system or part thereof [SAE ARP4761a].

Underwriters Laboratories developed the *ANSI/UL 4600 Standard for Safety for the Evaluation of Autonomous Products* to address safety considerations in fully autonomous systems while also addressing the reliability needed for ML [UL 4600]. The standard requires "a goal-based safety case" to ensure that an autonomous system is "acceptably safe for deployment" and is intended to work with existing standards and frameworks to support safety and TEVV aspects of ML-enabled and autonomous system development and deployment. For example, TEVV personnel can use the GQIM process, which we discuss in the Section 3.3.6, supplementally with UL4600's safety case approach to achieve system and mission-specific TEVV objectives. Section 3.3.6 also introduces the reader to some practitioner-level guidance for development and TEVV of ML-enabled systems, specifically at the ML model level.

The SEI developed *A Guide to Failure in Machine Learning: Reliability and Robustness from Foundation to Practice*, which provides theoretical concepts in reliability and robustness for ML models and some current techniques for reasoning about ML model reliability and robustness [Heim 2025]. Specifically, the SEI provides a definition of ML model failure and robustness-measurement techniques that are useful in practice. The guide also provides recommendations for measuring the effects of *sample uncertainty* and *prediction uncertainty.* Measuring the effects of these uncertainty terms provides empirical ML model performance degradation information that TEVV personnel can use to provide evidence for assurance to—and promote trust by—developers and evaluators.

Recognizing that the failure modes of an ML model—and, by extension, the failure detection methods within an item or system—are implementation and application specific, this guidebook provides some application-specific examples to contextualize the concepts at the ML-model level. For system-level concepts, recall that Section 3.3.4 discussed the concept of redundancy or "use of ensembles of machine learning components," and Section 3.3.3 provides guidance on considerations for "cognitive dissonance" in the PTR chain. Developers should use these concepts in a system's fault detection or health monitoring schemata.

The SEI has also proposed a formal framework for ML requirements validation [Turri 2022]. Beyond theoretical concepts, such as cost functions and expected or empirical errors, these artifacts provide techniques for evaluating ML models that can be useful during concept- and architecture-development phases or as part of the body of evidence in the V&V phases of a system. These theoretical concepts and evaluation techniques are all integral to designing a trustworthy ML-enabled and autonomous system with an appropriate level of fault detection and response.

3.3.6 Goal Question (Indicator) Metric (GQM and GQIM)

Building a strong assurance case involves providing a body of evidence that supports that the product does what it claims to do. At the core of any assurance case is the ability to trace the pieces of evidence back to the system requirements. At a high level, ML-enabled systems are no different: TEVV personnel and others must tie measures back to the requirements. Therefore, we must determine the right metrics to measure to build evidence for assurance.

There are many ways to measure ML algorithms, but TEVV personnel must determine which measure to use and what the proper values are on a case-by-case basis. For example, the OECD lists 123 technical metrics for trustworthiness, but there is no guidance about which to select for a given scenario or model [OECD 2024]. The question remains, therefore, about how to choose the right measure for the requirements.

There are a variety of techniques for developing software measurements derived from goals such as the Goal-Question-Metric (QGM) [Solingen 1999]. The SEI has developed a workshop based on an expanded GQM approach called Goal Question Indicator Metric (GQIM), which TEVV personnel can use to develop an assurance case. The presentation “GQIM and Assurance Cases” discusses how to do so and provides an example workshop report [Stewart 2015; Nichols 2023]. While this document is described as a workshop, the framework can serve independently as a reasoning method.

Figure 3 summarizes the GQIM process.

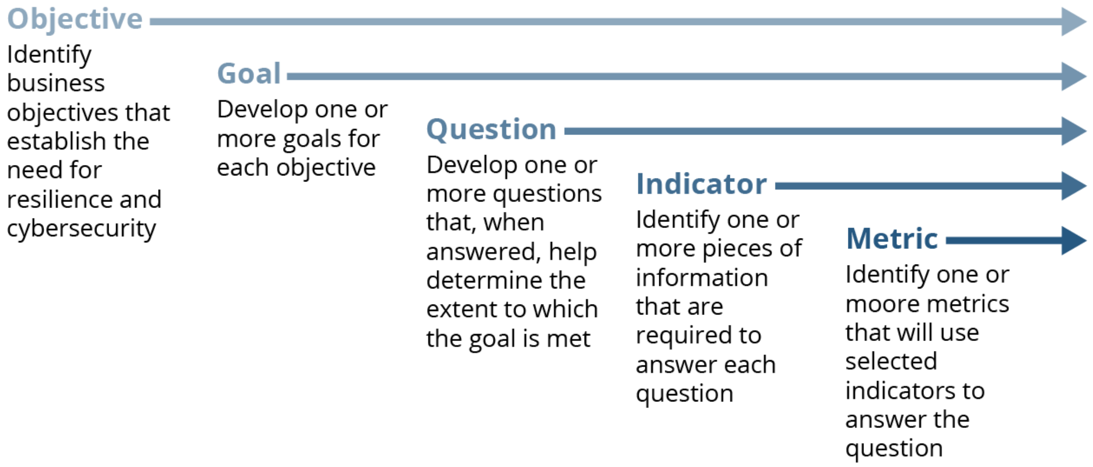
Objective
Identify business objectives that establish the need for resilience and cybersecurity
Goal
Develop one or more goals for each objective
Question
Develop one or more questions that, when answered, help determine the extent to which the goal is met
Indicator
Identify one or more pieces of information that are required to answer each question
Metric
Identify one or moore metrics that will use selected indicators to answer the question

Figure 3: GQIM Process Flow

TEVV personnel can use this method to build traceability between system requirements and ML measures and metrics. To provide an example of how to accomplish this task, we used GQIM to identify what metrics to measure using the objective of “Minimizing Civilian Harm.” The results of that exercise appear in Table 2.

Table 2: Example of How to Identify Metrics for the “Minimizing Civilian Harm” Objective

Objective: Minimize Civilian Harm [DODI 3000.17] • **Goal**: Maximize the ability of the system to identify civilians to prevent harm.	
Question, Indicators, and Metrics	**Question**: Do the training dataset(s) identify civilians encountered in the missions? • **Indicator**: contains labels and annotations that differentiate civilians from other classes ○ **Metric**: variability, balance, coverage • **Indicator**: Do the civilians represented by the dataset match the civilians in the target area? ○ **Metric**: compare model accuracy between two data sets • **Indicator**: data taken from appropriate operational context

	○ **Metric**: mapping of operation environment features to features in the collection environment
Question, Indicators, and Metrics	**Question**: Can the system identify civilians visually and rapidly? • **Indicator**: identifying all civilians ○ **Metric**: “high” recall on civilian class ○ **Metric**: “high” precision on civilian class • **Indicator**: time within limits ○ **Metric**: milliseconds to classification

This example defines metrics that the system might use to support civilian identification to prevent harm to civilians. This content serves only as an example and uses hypothetical metrics like “high.” In contrast, programs building real systems need to identify actual mission-specific values based on mission need, which may vary based on the specifics of the situation. For example, the metrics above are biased toward civilian identification and not toward enemy combatant identification or friendly warfighter identification. The timing metric example also must be contextualized within the overall response time. The same structure of the GQIM can be used for identifying those classes but should result in different confidence thresholds during operational use.

3.3.7 Perception, Targeting, and Environmental Difficulty

LAWS can be deployed in a variety of operational environments that present varying levels of difficulties for perception and targeting. Obviously, identifying the minimal set of specific operational scenarios greatly reduces the scope of development and operational testing whereas trying to address all combinations is infeasible.

There are many challenges for the PTR chain beyond common environment factors such as lighting, fog, rain, mud, and so on. Additionally, ML must deal with perception and interpretation factors in the scenes. Perceiving a single silhouette against a bright skyline requires a very different perception task from determining a single individual’s behavior from within a crowded urban scene in the dark. The following list offers several sample scenarios with different operational environments, resulting in different development and testing requirements:

- **Remotely piloted drones with targeting assistance**: In this case, the operators are responsible for all actual decision-making and only take input from the AI. This scenario doesn’t remove all responsibility from the AI in that it can suggest highly incorrect actions, but the operator acts as a safety.
- **Loitering aerial munition with targeting capabilities**: In this case, the munition is capable of a variety of advanced actions such as wayfinding and basic target identification. It is the responsibility of the operator to perform final target determination and to trigger the response. At that point, the system is self-guided for the remainder of its flight.
- **Fully autonomous ground system with lethal and non-lethal capabilities**: In this case, the device detects, nominates, prioritizes, and, under certain mission conditions, automatically responds, which in other conditions requires human approval.

For the sake of the ensuing discussion, we use three categories of entities that the system must identify: adversaries, friendlies, and neutrals (AFN). Neutrals are those entities not yet identified as adversaries or friendlies. For certain safety cases, it might be prudent to consider all neutrals as friendlies until such time as they can be confirmed as adversaries or friendlies, but for this conversation we will consider all three.

The following sections provide eight example questions to ask of the system when developing test cases to gauge the relative level of difficulties that tests can cover.

3.3.7.1 How difficult is it to identify the entities in a still frame?

The following list shows some examples of increasing difficulty.

- Simple structures with clear outlines, fixed locations and no motion. Note that this item is also the easiest for attackers to compromise using counter AI or AML.
- mobile devices with identifiable silhouettes or features, such as a ship, tank, quadruped, construction equipment, military aircraft, drones, and so on
- mobile devices with ambiguous features, such as trucks, SUVs, buses, civilian aircraft, and so on
- animals
- humans

Development teams should note the following key points:

- Motion tracking is often different from static object detection and can require combinatorial testing approaches.
- Inanimate systems with highly discrete features are obviously easier to identify.

3.3.7.2 What is the composition of entities within the scene?

The following list shows some examples of increasing difficulty.

- only adversarial targets
- mixed adversarial and friendly targets but simple perception difficulty such as ships
- complex, mixed environment of adversarial, friendly, and neutral entities of varying difficulty types

Early systems and some guided munitions may be limited to environments where only adversarial entities are expected to operate or where adversarial entities are expected only when the device is operational. An example of such a scenario might involve a case of a guided munition in which all perception and targeting decisions have been made before the munition is activated.

3.3.7.3 What confounding qualities exist in the scene?

Damage, smoke, camouflage, and counter AI may occur accidentally or intentionally, but all these eventualities must be accounted for during training and testing. Battle damage assessment has been an area of study for some time but is still an ongoing problem for ML. Traditional camouflage and counter AI can have similar effects where they work to misclassify a target.

3.3.7.4 How many classes of each AFN type are to be identified?

In some cases, it may be desirable for the device and datasets to have different subclassifications within a higher-level classification. For example, within the tank classification, it is important to separate different types of adversarial tanks or ships for different types of responses. Obviously, within human targets, this discrimination is significantly more challenging. TEVV personnel must be mindful of the complexities of trying to test a device that is intended to finely differentiate between subclassifications of people.

3.3.7.5 How many entities are expected to be in the scene?

As the number of entities in the scene increases, independent of the overlap, the speed at which many object detectors perform decreases. Quality suffers in systems with real-time deadlines that must contend with numerous entities because the system struggles to meet those deadlines. This scenario results in decreased confidence in the system.

3.3.7.6 What mixture of entity types is expected in the scene?

The ratio of the different AFN types can affect the types of metrics developers or TEVV personnel choose for measuring system requirements. An operational environment populated only by adversaries, such as a surface fleet, involves much simpler decision-making than a highly mixed urban environment composed of equal numbers of each entity type.

3.3.7.7 Does AFN determination require tracking or assessment of behavior?

In some situations, silhouettes or other similar static factors are not enough to identify adversarial behavior to support accurate target identification and response determination. Before prosecuting an attack, the system might need to observe an entity for a certain amount of time to acquire enough identifying samples from multiple frames to reach a certain confidence threshold. Alternately, behaviors can be determined by motion across frames such as with keypoints or behavior models that require several frames.

TEVV personnel will need to highly scrutinize requirements and thresholds for any determinations that require a significant amount of time to make. The system might express such determinations in terms of confidence windows where the system has “accumulated” enough evidence over a certain period for accuracy and to establish the absolute minimum thresholds needed for determination.

3.3.7.8 How much trajectory crossover is allowed?

During PTR activities, there are certain situations where the system might lose visual surveillance of an entity that it is tracking. At some point, the system might recover tracking of the entity. We refer to the period during which the system can’t track the entity as a period of crossover.

Entities may cross over for varying lengths of time if they are moving quickly or if they become obfuscated. For example, objects might behave like players in a basketball game where they move with extreme agility and can change directions and become obfuscated. Or, they might behave like billiard balls where the entities can change direction due to collisions making tracking difficult and producing periods of crossover. System designers must establish how the system will perform crossover detection, especially if the entities are highly mobile.

3.3.7.9 Are IFF detectors employed?

If Identification Friend or Foe (IFF) technology is employed, it can significantly assist in identifying friendly entities but may not improve the positive detection of adversary or neutral entities. Adversaries can spoof or steal IFF, so the system should use it as just another indicator within the overall PTR chain.

When systems use IFF, they should provide details about the types of classes it applies to. For example, an IFF designed for a vehicle but identified on a human should be considered suspect and raise concern.

3.3.8 Levels of Autonomy from Human Control to Full Autonomous Systems

There is a lot of variation between devices in terms of the amount of AI and ML they use as well as how autonomous the device is and what degree of control an operator might have over the functions of the AI and ML. For example, some "simpler" systems, from an AI perspective, are remotely controlled or piloted by an operator, and ML might serve only in more "minor" assistive flight tasks through recommendations that the operator can ignore. More "complex" systems might support greater autonomy for operations such as navigation and guidance.

Guided munitions have been around for a long time, and many traditional electro-mechanical systems increasingly incorporate more and more software, from the use symbolic AI to adoption of ML. Over time, as the level of autonomous decision-making increases, the human takes a more supervisory role. At the furthest end of the spectrum, we envision fully autonomous devices navigating entirely on their own and choosing and engaging targets.

When developing requirements and tests, it is critical to understand the extent of human control. Clear and detailed descriptions of how the user interacts with the LAWS, which decisions the LAWS makes, and under what conditions LAWS requests support are critically important to provide effective operational test environments. Additionally, TEVV personnel must define what feedback mechanisms the system uses in conjunction with its decision-making processes. Chapter 2 of the *Reference Architecture for Assuring Ethical Conduct in LAWS* discusses how human trust is deeply dependent on understanding how the system makes decisions. Human trust is also dependent on getting understandable feedback about what state the machine is in and understanding when operators are required to interact with the system [Mellinger 2025].

SAE has defined six different levels of autonomy for self-driving cars [SAE 2021]. Other standards such as MIL-STD-882E also define autonomy in their "Software Control Categories." TEVV personnel can use these categories as guidelines when defining the decision-making scenarios for requirements and testing. Different operational modes might require different levels of decision-making, and TEVV personnel should define these modes separately. For example, a machine operating in reconnaissance mode may only require the user to set areas for search, but in an urban environment the user may be required to provide more precise waypoints and even directly assist in navigation. Additionally, as mentioned above, the complexity of the environment can have a direct impact on the level of operator interaction.

3.3.9 Timing

Requirements and scenarios should specify end-to-end timing needs and consider such factors as human-response time, system-response time, and situational demands. In any autonomous system, the chain of perception to reaction must occur quickly enough that the final action is relevant to the initial input. Researchers have extensively studied these issues in safety-critical systems such as avionics.

LAWS add the dimension of user-response times when they involve human-in-the-loop or human-on-the-loop. Given a particular situation, we must determine whether it is reasonable for a system to determine the need for human approval, send a message for human approval, or wait for approval and respond appropriately. For certain situations, the system might require preapproval, such as when the user nominates certain targets for automatic response rather than on a case-by-case basis.

In general, human-reaction times are on the order of 250 milliseconds from visual stimulus when ready. Network latencies can last between 7-25 milliseconds for a single message, without interference. In a PTR chain consisting of multiple ML components that need to communicate with a human about whether to engage *in direct response to an action*, it quicky becomes apparent that a human may not have time to approve engagement in demanding situations.

When evaluating mission scenarios and requirements, it is therefore important to consider all the following:

- sensor data acquisition time
- PTR chain execution time
- communication time to operator
- the operator’s readiness to respond combined with their response time
- communications time back to the machine
- time to engage

Any engagement requirements must specify realistic timing so that developers can design the system to work within that time. For example, if a requirement specifies that the operator must provide approval within 200 milliseconds of detection, it is highly unlikely, even with perfect processing speed and perfect network speed, that an operator would be able to respond within that time frame.

3.3.10 Operator Task and Cognitive Loading

Human-reaction times are affected significantly by the *cognitive load* involved in the decision, which includes both the inherent complexity of the task as well as the way the information is presented. Tasks with a high cognitive load take more time, and any sort of distractions, such a distrust of a system, can lengthen this reaction time.

When specifying mission context and requirements, TEVV personnel should address cognitive load, tasks, and distractions accordingly. When doing so, it is important to remember that, even though warfighters are trained to react in challenging environments, they are likely to have had less training on human-machine teaming (HMT) with respect to LAWS than training to work with human counterparts.

TEVV personnel must evaluate whether requirements for operator interaction present decision-making to the user with sufficient information and feedback to elicit appropriate responses with respect to cognitive load. For example, it is common in object-detection systems to show bounding boxes that identify objects with confidence scores. TEVV personnel should check whether these display elements use well-understood and common mechanisms and whether they provide enough information to reduce the decision load. In the study "Measuring Trust: Concept Testing and User Trust Evaluation in Autonomous Systems," we identified that the participants preferred color coded boxes with confidence scores on a high-medium-low scale [Hale 2025].

TEVV personnel should also find whether any studies exist that describe relevant cognitive load measures that could assist in testing. For example, NASA task load index (TLX) is a tool for collecting workload assessments [Hart 1986]. TEVV personnel might consider whether this sort of measure has been used to understand which tasks require or may require more cognitive load, and under what conditions. These conditions can be used to create additional tests.

Additionally, tools like the STS can be used to evaluate the effectives of these interfaces on operator trust.

3.3.11 Dimensions of Autonomous Decision Making (DADs)

The document "Dimensions of Autonomous Decision-making" identifies 13 categories of risk elements to help reflect on risk in the use of autonomous systems [Stumborg 2021]. The paper proposes the creation of a Joint Autonomous Risk Elements List (JAREL) based on the DADs provided in the paper. At the time of this writing, a JAREL or similar risk list has not been developed. However, until a list is created, TEVV personnel can effectively use the questions in the DADs to identify requirements and then to identify metrics.

DAD #10, titled "Test and evaluation adequacy," is directly applicable to TEVV. The paper describes it as follows:

> *Will/did the IAS test and evaluation procedure reflect the breadth, depth, and complexity of the contemplated operational environment to test and evaluate the system attributes unique to the use of autonomy technologies to the greatest extent practicable? [Stumborg 2021]*

These considerations directly refer to the testing process and how TEVV personnel can perform the testing. Working through some of those considerations can result in significant project impact. The following question from the DADs provides an example of how they can help impact testing: “Did the IAS test and evaluation procedure receive an already-trained IAS, or is the IAS to be tested also to be trained in the replicated operational environment?” [Stumborg 2021]. If the IAS is to be trained *during* testing, then a completely different approach is needed than if the IAS had already been trained beforehand. While we don’t expect such a scenario to arise during practice, there may be times when the system did not have enough access to accurate operational data and therefore relies on training data acquired during testing. Developers and TEVV personnel can overcome this issue by using an iterative approach like MLOps, where the device is trained and perhaps iterated over during testing. To employ such a method, training pipelines, processes, equipment, and personnel must be available during testing to train and tune such models. Adopting this process can significantly increase the cost and time for development and testing.

The other DADs can inform requirements development and test plan production. For example, DAD #5, titled “Command and Control,” asks the following question: “Is the IAS prohibited from initiating operation in the absence of a control link to a human operator?” [Stumborg 2021]. The results of working through this question can lead to very different testing paths and might accrue considerable time and risk. In DDIL environments, defining “absence of control link” depends on how absence is defined. Defining the term might mean working through several questions, such as the following: Is there a particular data bitrate that is required to support information to the operator and return control data? What control acknowledgment mechanisms are provided to guarantee delivery of critical information? When delays occur, what responses times are required on a per command basis? Different user commands may require different response measures based on associated risk.

As mentioned above, we hope some method like GQIM will become available in the future to provide a detailed set of examples and guidelines for how to create or evaluate such requirements, and we hope that such a method will be published as a standard document.

3.3.12 Values Criterion Indicator Observables (VCIO)

The VCIO model provides a way to describe and map socio-technical values into system observables. The intent of the model, as described in the specification, is to establish if “it is possible to describe whether a product adheres to specific values and can be trusted” [VDE 2022]. The model iteratively progresses from values, through criterion, to indicators, and finally observables. It is conceptually similar to the GQIM method, but the end focus is a little different. GQIM tries to focus on continuous measures where VCIO usually works on ordered measures.

In the book *From Principles to Practice: An interdisciplinary framework to operationalise AI ethics*, the authors use the VCIO process to provide some structure for discussing AI ethics such as transparency or accountability [Krafft 2020]. The book uses a variety of numerically ranked observables where each higher-numbered observable has more stringent criteria. For example, 0 is associated to “No,” 1 is associated to “Data with mission observations,” and 2 is associated to “Complete data.” They then combine these numbers across multiple observables and criteria to provide an A-through-E grade for each value. Finally, they produce a graphic that shows all values to provide an "AI Ethics Label" inspired by the energy efficiency label. Structured assessments like these improve understandability, and they make it easier to compare different assessments. When possible, TEVV personnel can use these kinds of structured assessments with approaches like GQIM—which we discussed in section 3.3.6—to improve their processes. While this document does not directly map to the RAI principles and tenets, developers and TEVV personnel can make use of much of the analytic thinking and detail the authors provide to inform any requirements development or evaluation method.

3.3.13 Emergency Stops

Automated systems can make decisions and execute actions without operator intervention. In the context of LAWS, these decisions include choosing where and when to target and engage entities. These systems make decisions using the symbolic logic embedded in the system's design and code. However, introducing ML models into the decision-making pipeline also introduces new dangers in the system's operation.

For example, the decisions made by a LAWS that use object detection and classification models to inform its PTR chain depend on how its ML models classify input data from its various input sources (e.g., FMV, lidar, etc.). As noted in Section 3.3.9, the process of receiving sensor data, classifying it, determining a response, and executing a response should take no longer than tens or hundreds of milliseconds. If the system's ML models accurately classify objects it detects through sensor data, designing this decision pipeline to unfold as quickly as possible improves the system's ability to successfully perform safety-critical tasks by enabling faster responses or by providing more classification attempts in the same time period. However, such would not be the case if the system's ML models inaccurately classify an object, which may happen if the system encounters an object that it's never encountered before. If the system inaccurately classifies an object and the system's speed is insufficient, then the system may execute a lethal response incorrectly, which it would have avoided had the system had more opportunity to make additional classification attempts. This outcome can occur in fully automated systems when the process does not run quickly enough to avoid these unintended consequences.

The example above showcases the need for developers to design a way for LAWS to mitigate unintended consequences if their ML models underperform. One way to provide these mitigations, as mentioned in Section 3.3.2, is to implement design patterns that proactively limit system behavior in certain contexts. In situations where unintended consequences were not or could not be prevented, LAWS must have mechanisms that enable their operation to be altered, interrupted, or stopped altogether. One such mechanism is an emergency stop, or a "deactivation switch." If the system has made undesirable decisions, operators—or even the system itself—must be able to activate such a mechanism to prevent the system from incurring further damage. Often, these mechanisms occur as operator manual override functionality.

Requirements for LAWS should include emergency stop mechanisms, such as deactivation switches and manual overrides. The use case and deployment context will inform the amount, type of, and physical and logical locations of emergency stop mechanisms. Therefore, developers should consider both the ML components and the intended system deployment context when writing requirements for emergency stop mechanisms.

3.3.14 Automated Decision Transparency and Traceability

Though decisions made by LAWS are automated and generally occur within milliseconds, humans should be able to retroactively review what decisions the system made, as well as the input it used to make each decision. This ability to review system decisions adheres to the traceability principle of the DoD's Ethical Principles for AI, which describes it as follows: "AI capabilities will be developed and deployed such that relevant personnel possess an appropriate understanding of the technology, development processes, and operational methods applicable to AI capabilities" [DoD 2021a]. Traceability provides several major benefits, but for the purposes of this document, we want to highlight the following three:

- If the system makes an incorrect decision, traceability helps development teams diagnose and fix the offending system components.
- Transparency in the decision-making process engenders operator trust in the system's ability to make correct decisions.
- Traceability during system operation enables operators to identify potential issues in decision-making processes (e.g., the PTR chain) and prevent these issues from causing unintended consequences.

Ideally, every decision the system makes, as well as the data used to make that decision, should be logged with timestamps and made available for post-operation analysis. These measures enable review of the system's overall performance with respect to its system-level decision pipelines. Even further, if the system uses ML models, it should log inference results from all models with timestamps and tie them to the input data it used for each inference. This logging is required to perform post-operation evaluation of the system's ML models, and it should occur in addition to all traditional logging mechanisms for LAWS.

The system's context will heavily influence the number and type of transparency mechanisms required by the system's design. For systems with restricted resources, logging every decision and all input data may not be possible. Such extensive logging may even impact operation if there is insufficient computational power to support it. In these circumstances, requirements should indicate a minimum set of data points and decisions that the system must log or surface to operators during system use. These requirements should target the highest impact decision pipelines, such as PTR chains, and focus on the core components of the pipelines, such as object detection and classification ML models.

While logging for post-operation analysis should be as comprehensive as possible, Section 3.3.10 demonstrates that developers should consider operator cognitive load when determining what information to surface to the operator during system use. Operators should receive as much information about the highest impact decisions as possible without distracting operators from their mission or creating cognitive overload. Developers should consider decision transparency when they document requirements for who will be using the system, how they will be using it, and how the system will deliver decision information to operators (e.g., through a user interface, audio comms, haptics, etc.).

3.3.15 Replicable Tests for ML Components and Use Cases

Replicable tests serve several purposes. First and foremost, they provide evidence of the safety and performance of your system, and they show in an auditable way that it meets all the requirements it claims to meet. For this reason, all requirements should correspond to a test that can verify them so that you can show that the whole system meets requirements, as we recommended in Section 3.1.7. A traceability matrix, which maps each requirement to a test (or multiple tests) is a powerful tool for ensuring coverage and therefore correct operation. Auditors can later review these tests to determine if they provide evidence that the system meets requirements, examining both the resulting metrics as well as the data used to generate the tests.

Secondly, replicable tests are a mechanism for assuring any updates to the system are compliant with the system's requirements. These tests allow TEVV personnel to evaluate any new model, with a new architecture or new set of training hyperparameters, against the previous set of tests to ensure that it continues to meet requirements. It also gives TEVV personnel a way to compare the updated model to the existing model's performance. Developing these tests can also provide utility when closing the loop between model development, data collection, and evaluation because they can help identify areas of system weakness where more data or better image augmentations are needed.

Not all system requirements can be represented using typical performance tests and metrics, such as latency or model size. Where possible, TEVV personnel should find ways to automate these tests as well, even if when doing so is as simple as loading the model and checking the size in memory. For these tests, the exact data may not matter, but TEVV personnel should still note the results. If an aspect of the system changes, such as image resolution, developers will be able to see and isolate potential impact on system performance.

The number and range of tests depend heavily on the system's use case. Both requirements and the data itself can vary between use cases and contexts. For example, LAWS with separate modes for an open field of combat as opposed to those that function in a contested urban area where civilians may be present should each have an independent set of tests. Showing that precision and recall function as per requirements for the system deployed in an open field of combat does not mean that that system will perform adequately when civilians are in the mix. Specifically, by only looking at open field data, you would not be able to quantify the false positive rate of detecting civilians as combatants. It can be useful to repeat requirements in each context, so that each context contains all the requirements needed to operate. Doing so ensures that TEVV personnel test all high-level system requirements in each context.

4 Data Collection, Curation, and Management

This chapter discusses the integral role of data in machine learning- (ML)-enabled systems (MLES), and specifically in lethal autonomous weapons systems (LAWS) in use by the Department of Defense (DoD), where the trustworthiness of such systems is crucial. The chapter provides guidance for personnel who perform testing, evaluation, verification, and validation (TEVV) of these systems to use data as a first litmus test of a system and as an opportunity to understand the nuances of subsystems that might make subsequent use of the data.

This chapter focuses especially on the data for training, validating, and testing ML models to clarify the relationship between data characteristics and the correctness and reliability of MLES. In data science and the field of ML, a guiding principle is that an ML model is only as good as the data it's trained on, so if bad data goes into the model, bad data is bound to come out. As we outlined in Chapter 1, generating trustworthiness in ML systems is fundamental to ensuring operators and teams work toward maximum efficiency. For that reason, the data an ML model produces should not only be accurate, but operators should be able to trust it. Importantly, the Johns Hopkins University Applied Physics Laboratory states that "data curation is not just a preparatory step for building an AI-enabled system but an opportunity to promote trustworthiness in that system" [Clemens-Sewall2024]. From a TEVV perspective, it is also a chance to confirm the trustworthiness of an artificial intelligence (AI) system. This chapter provides guidance for TEVV personnel to evaluate data to promote such trustworthiness.

4.1 Recommendations for Data Management

The following recommendations provide guidance for TEVV personnel to prepare data adequately and take precautions to promote a trustworthy system.

4.1.1 Assess Data Management and Datasets

High-quality datasets can be compromised by low-quality data management. The importance of data management is demonstrated by its inclusion as the topic of a section in DoDM 5000.101 and DoDI 5000.98. TEVV personnel should be aware of the seven goals outlined in those documents, which stress that data management should strive to make data visible, accessible, understandable, linked, trustworthy, interoperable, and secure. Those seven goals are represented by the initialism VAULTIS, and they comprise the DoD's new data strategy that it introduced in 2020 [Hicks 2023]. That document also provides a checklist for performing an assessment of how TEVV personnel are using data management and whether it adheres to VAULTIS.

4.1.2 Incorporate Data Provenance into Dataset Evaluation

TEVV personnel should validate that the entire data lifecycle is trustworthy through proper documentation, application of calibrated rationales, use of ethical practices, and alignment with the project's mission. To establish these practices successfully, Section 3.8.a.1 of DoDM 5000.101 promotes the development of data cards, and DoDI 5000.98 establishes the requirement of a transparent data pedigree. Both documents express the need for datasets and testing and evaluation (T&E) data to be "visible, accessible, understandable, linked, trusted, interoperable, and secure" [DODM 5000.101; DODI 5000.98]. These qualities foster a trustworthy system and are promoted by data provenance.

4.1.3 Establish Data Governance and Compliance

Projects should comply with the guidance and requirements provided in current applicable governance documents. As of this writing, we have identified four such documents—DoDI 5000.98, DoDM 5000.100, DoDM 5000.101, and DoDD 3000.09. DoDI 5000.98 and DoDM 5000.100 both contain sections on data management with which TEVV personnel must comply [DODI 5000.98; DODM 5000.101]. DoDD 3000.09 specifies that systems should be tested and evaluated to ensure that their autonomy algorithms can be rapidly reprogrammed on new data [DoDD 3000.09]. DoDM 5000.100 contains the most detail, describing the role that data governance must play in T&E Master Plans (TEMP) and T&E Strategy, with some specific requirements for data and data-source management [DODM 5000.10].

DoD guidance and requirements for implementing data governance is sparse. However, as the DoD moves to adopt AI at a larger scale, it will release more documents with further guidance, and TEVV personnel must be on the lookout for new publications to ensure they are meeting the most current requirements for data. As this discipline evolves over time, TEVV personnel must work to stay up to date with DoD directives that directly affect data management standards for LAWS.

4.1.4 Evaluate Data for Its Current Quality

Datasets can have different levels of quality that dynamically shift over time due to changes in prioritization, fluctuating conditions, and many other factors [Hicks 2023]. Quality datasets tend to score high in dimensions of accuracy, completeness, conformity, consistency, uniqueness, integrity, and timeliness. The DoD's *Data, Analytics, and Artificial Intelligence Adoption Strategy* provides a useful, high-level checklist that can function as an initial assessment [Hicks 2023]. When evaluating LAWS, TEVV personnel should leverage similar tools and checklists to ensure the quality of datasets used to train and test LAWS are of sufficient quality to meet DoD required standards.

4.1.5 Validate That Datasets Provide Full Coverage of the Operational Design Domain

TEVV personnel should create testing to validate that datasets provide full coverage of the system's operational design domain (ODD). TEVV personnel can meet this recommendation by using metadata to track key aspects of the data and by performing analysis of the metadata to understand how much data support there is in various regions of the ODD. Such regions should also include combinations of conditions, like "raining" and "at night."

Ideally, developers would have at least some data that represents all possible circumstances that the system's ML models may encounter. In such a case, the model would be trained on data equivalent to deployment data and would respond to all situations appropriately. Due to the sheer size of the data and the countless possible combinations of circumstances that a deployed system might encounter, accomplishing this ideal scenario is infeasible. This is why an ML model should be trained on enough data to enable it to generalize about situations that it did not "experience" in its training. TEVV personnel that are testing LAWS that use ML models should, therefore, be rigorous in diversifying their testing dataset. TEVV personnel may find scenarios that developers did not consider during training, which provides an additional chance to identify and address weaknesses in model performance.

4.1.6 Test That the System Accurately Detects Data Drift

Even if ML models perform well in their initial deployment environment, the characteristics of and trends in the data they encounter in the field will likely change over time. This circumstance is especially true for models in systems like LAWS that are subject to ongoing variations like changes in environmental conditions, new deployment locations, and numerous human factors. Such changes in data over time are referred to as "data drift."

Data drift is inevitable in most situations and prevention is not practically attainable. To help mitigate potential loss resulting from data drift, TEVV personnel should verify that the LAWS under testing can detect data drift. TEVV personnel should purposely expose the system to data and conditions that are outside of, but adjacent to, the scope of the LAWS to review how the system reacts to data that falls outside of its initial training domain. Data-drift detection acts as a proactive way to prevent poor model performance during operation and provides a mechanism for improving operator trust in system performance and control over the system.

4.2 Commentary on Data Management

This section provides in-depth explanations for the recommendations and observations listed above.

4.2.1 Data and Ethical Principles for AI

The data used to train an ML model is the foundation from which the model draws when it contributes to decision pipelines in LAWS. Flaws in the training dataset will manifest as flaws in the model's—and likely LAWS'—operational performance. Unless TEVV personnel specifically test for them, these flaws may not become apparent until a related edge case occurs during operation. Given the dangers of flaws in the operation of LAWS, the testing and evaluation of LAWS should include steps to identify as many of the model's hidden flaws as possible. It is impossible to guarantee that TEVV personnel will be able to account for all edge cases, but the DoD's Ethical Principles for AI document can act as a reference to infer general areas, and implicitly general flaws, on which to start focusing testing efforts. Chapter 2 of our companion guide *Data Curation for Trustworthy AI* provides detail on the challenges and approaches for characterizing data [Clemens-Sewell 2025].

Responsible system design and operation is difficult to test and measure due to its subject nature. TEVV personnel should frame the task of testing for "responsibility" as testing to verify that system designers and developers did their due diligence to reduce, as much as possible, the potential for the system to do unintended harm. From this perspective, TEVV personnel must question the extent to which designers and developers responsibly selected and used training data for this specific system. For instance, TEVV personnel should verify to the best of their ability that the data used to train the system's ML models is sufficiently relevant to the context in which the system will be deployed. They should also evaluate training data to ensure that developers did not unintentionally or irresponsibly use sensitive data, such as restricted government or data or personally identifiable information (PII), to train the system.

Testing the system for equitability most often takes the form of testing for the existence of bias in the system's ML models. Though usually unintentional, models often contain unintended biases due to a lack of diversity in the training dataset. Such bias will cause the models to influence the LAWS to favor certain actions or decisions in certain situations, regardless of whether that action is ideal or correct. Therefore, TEVV personnel must actively test the system's ML models for the existence of biases. If access can be provided to the initial training data, then various techniques can be used to understand the data characterization and potentially adjust the dataset to compensate for the details. Chapters 6 and 7 of our companion guide *Data Curation for Trustworthy AI* provide more information [Clemens-Sewall 2025].

TEVV personnel should also test ML models using diverse datasets that represent various demographics, conditions, contexts, and potential outliers. Such testing involves identifying edge cases and deployment context variants and gathering or generating additional testing data to specifically test against these anticipated conditions. This testing also provides insights into the system's governability by providing opportunities to detect and avoid unintended consequences that may arise from issues in the training data. The AI Fairness 360 tool is an example of a tool that TEVV personnel might use to actively test and mitigate biases in datasets. It is available through an open source library, and it can detect and mitigate bias in ML models throughout the AI application lifecycle [Trusted-AI 2025].

As outlined in the DoD's Ethical Principles document, AI capabilities should be "developed and deployed such that relevant personnel possess an appropriate understanding of the technology" [DoD 2021a]. In other words, the DoD stresses that AI capabilities should provide mechanisms that enable their users to understand what actions they are taking and what motivated those actions. With respect to data, testing for traceability starts with verifying that all data sources, design processes, and methodologies are fully documented and understandable to relevant stakeholders. TEVV personnel should check that there is documentation for every dataset used to train ML models, and they should scrutinize its quality. In addition, TEVV personnel should review records of dataset provenance to verify that all data is traceable back to its source, and they should verify the documentation of all data sources, including when, where, and how data was collected, as well as any subsequent processing steps. Using these methods to verify data traceability also provides a way to evaluate the reliability of the datasets and the data management processes that developers used during system development.

In addition to using the DoD's Ethical Principles for AI document, TEVV personnel should consult other responsible AI frameworks for additional guidance. As an example, the Institute for Ethical AI & Machine Learning published *The Principles of Responsible ML*, which provides a list of eight principles that TEVV personnel can apply directly to LAWS to test if the system was developed responsibly [Institute for Ethical AI & Machine Learning 2025].

4.2.2 Data Splitting

Data splitting is the practice of partitioning a data set into two or three subsets with one part for training the model, one part for validation the model, and the optional third part for testing the model [CDAO 2024]. There are a variety of techniques for constructing these subsets such that each properly represents the operational domain distribution. The CaTE companion guide on *Data Curation for Trustworthy AI* provides details on general splitting techniques, how to split while maintaining distribution diversity, resampling for class balance, and guidance for the practitioner on splitting [Clemens-Sewell 2025]. The document specifically emphasizes the negative effects of overlap between training and test data splits as well as the potential for misleading test results when using data outside of the intended system's operational design domain.

4.2.3 Verifying Dataset Relevancy

Data collection is the first step in the process of training an ML model, and it provides the first opportunity for TEVV personnel to focus on data provenance and ensure that the data comes from quality sources that match mission scenarios and goals. The quality and appropriateness of the data collected will propagate throughout the rest of the process. We have found that there is little to no guidance available for TEVV personnel to evaluate the data-collection process. However, based on our team's expertise, government supplied guidance, and the evidence we've gathered from real-world ML projects and research, we've noted effective practices across most use cases. We share some of that information in this section.

Though superseded by the *DoD Data, Analytics, and Artificial Intelligence Adoption Strategy*, the *DoD Data Strategy* provided guiding principles for data management and still contains relevant valuable guidance. In particular, Section 2.2.7 of the *DoD Data Strategy* explains that TEVV personnel should verify that training data was "fit for purpose," or, in other words, they should validate that the data used to train ML models for a LAWS is a suitable proxy for the data that a LAWS will see during deployment [Norquist 2020]. Suitable datasets should fairly represent the variance in environmental phenomena that the LAWS is likely to encounter in the domain of the use case. TEVV personnel should therefore verify that the data used to train ML models in LAWS represents the system's target domain as accurately as possible.

Validating the relevance and quality of a dataset is a difficult problem. The CaTE companion guide on *Data Curation for Trustworthy AI* discusses several different methods developers can use to align their datasets with the system's true operational distribution. These methods include, among others, leveraging domain SMEs, asking appropriate questions regarding the operational domain, and applying algorithmic techniques to find differences between the true distribution and the distribution found in the TEVV datasets [Clemens-Sewell 2025]. TEVV personnel can leverage these same techniques to assist with gathering appropriate validation datasets and evaluating the quality of the datasets used during model training.

The types of ML models and architectures employed by the system can also influence data-collection requirements. Different model types require different types of data, and TEVV personnel need to be able to identify if developers used incorrect data types or erroneous data annotations for model training purposes. Models that use different learning types offer a good example to illustrate this point. For instance, modules that use supervised learning rely on pre-labeled datasets to teach algorithms how to interpret data and predict outcomes. However, models trained with unsupervised learning techniques only require relevant data and do not need labels. Finally, models that use reinforcement learning techniques learn by interaction with an environment via a reward function, not from a static dataset. Each learning style requires different methods of data collection, cleaning, and labeling (or not labeling). TEVV personnel should therefore validate that the methods used to collect, clean, and enrich each model are appropriate given the model's type and architecture. The CaTE companion guide on *Data Curation for Trustworthy AI* provides example questions that TEVV personnel can ask about a system and its components to assist in defining what kind of data collection and labeling processes are appropriate for the system [Clemens-Sewell 2025].

The use of data cards—which aid users in understanding the details and context of datasets—is a practice that has gained recent popularity, and it is becoming widely adopted [Pushkarna 2022]. Though there is a lack of standardization in the greater data-management community, data cards can help TEVV personnel identify the relevancy of a dataset. If systems under testing are delivered with data cards for datasets, TEVV personnel can directly compare the fields and metadata in the data cards to data-centric system requirements. Additionally, TEVV personnel can leverage tools such JATIC Gradient to generate custom data cards to compare as-is data to user-supplied data for data verification [CDAO 2025]. Tool-assisted generation of custom data cards enables TEVV personnel to customize a set of metadata that best identifies the dataset characteristics they want to focus on when analyzing dataset relevancy.

Often, systems use ML models trained on a dataset that are supplemented in some way by synthetic data. Synthetic data is artificial data that developers create manually or extract from an artificial or simulated source rather than gather from the real world. When synthetic data is of sufficient fidelity, it can supplement training data for classes or contexts that developers were unable to capture naturally. However, the effects of using synthetic data to train models are largely unstudied, and determining how well synthetic data generation techniques succeed in approximating real operational domains is still an open field of research [Clemens-Sewell 2025]. TEVV personnel should be skeptical of all datasets that include synthetic data. Synthetic data is appropriate for use in many situations, but when performing TEVV of LAWS, TEVV personnel should inspect training datasets for the use of synthetic data and scrutinize the reasoning behind the inclusion of such data. Further, TEVV personnel should avoid relying on synthetic data as a primary source for testing model performance. That said, it may be appropriate to use synthetic data to assist with the testing of edge cases for which data was difficult or impossible to capture naturally.

4.2.4 Data Augmentation for Training and Testing

In the development of robust ML models, data augmentation serves as a pivotal technique to enhance the diversity and volume of training datasets. The essence of data augmentation lies in its ability to artificially expand the dataset by introducing a range of realistic modifications to existing data points. Especially in scenarios where data may be limited or overly uniform, data augmentation helps prevent overfitting and can improve model performance across varied real-world conditions.

Augmentation techniques vary significantly depending on data type. In image-based applications, transformations such as rotations, scaling, normalization, and brightness adjustments enable models to recognize objects under different conditions and perspectives, thereby improving their adaptability. Similarly, in natural language tasks, techniques like synonym replacement and back translation help models grasp the nuances of language, ensuring robustness against linguistic variations. Audio data, too, benefits from augmentations such as noise addition or pitch alteration, preparing models to handle real-world auditory variations effectively.

In some cases, developers include augmented data in the training dataset as a form of synthetic data. In other cases, developers augment data during model training using various automated data manipulation techniques. In either situation, TEVV personnel must validate that the chosen methods align with the LAWS' intended operational environment. For instance, augmenting image data with noise might be practical for surveillance systems operating in visually complex scenes but less so for those operating in controlled environments. Equally important is determining if the augmented data introduces any biases that could skew the model's learning. TEVV personnel should verify that the modified data still accurately reflects real-world conditions and doesn't distort the model's perception of the environment. Part 2 of our companion guide *Data Curation for Trustworthy AI* provides a detailed discussion of data modification and approaches to ensure the resulting datasets retain their fidelity and relevancy while promoting the intended properties [Clemens-Sewell 2025].

Data augmentation can provide additional benefits in the model-testing process. While not a replacement for an actual domain-relevant validation dataset, TEVV personnel can apply the same methods for augmenting data during model training to supplement the dataset they will use to validate model performance. Introducing augmented data into the validation dataset is one method for testing model robustness. However, TEVV personnel should leverage data augmentation cautiously. There is no guarantee that systems that perform well on augmented data will also perform well on the data being represented by the augmentations. Additionally, if using data augmentation techniques during testing, TEVV personnel should not leverage the same techniques used during training. Doing so may cause the validation dataset to overlap with the training dataset.

4.2.5 Data Labeling

Data labeling is a task that involves labeling or tagging data points to prepare the data to train ML models. This process is necessary to produce datasets for supervised learning pipelines. Development teams usually perform data labeling after data collection, but sometimes they can also do it during the data-collection phase. Data labeling can be costly, time consuming, and arduous, and this step in model training is therefore often automated, outsourced, or rushed. Because of these known challenges, TEVV personnel should thoroughly inspect the labeling processes used by developers, seeking specifically to identify areas where developers may have cut corners or where they may not have performed their due diligence.

Traditionally, the method for data labeling involves people, rather than machines, applying labels based on their opinion and expertise. However, people make mistakes, and different people's opinions or interpretations of a label may often differ. Data labeling tools such as CVAT are useful for speeding up the process through automation [CVAT 2025], but automated tools like these are also capable of (and prone to) making mistakes. Because both people and automated tools can make labeling mistakes, TEVV personnel should be diligent to validate that labels are applied consistently and accurately across the training and validation datasets. Tools such as JATIC RealLabel can assist in identifying problematic or incorrect labels [CDAO 2025].

5 ML System Design

Machine learning- (ML)-enabled lethal autonomous weapons systems (LAWS) include ML components that make predictions about the input information and apply additional logic to generate autonomous actions based on interpretations of those predictions. Currently, ML technology is not sufficiently advanced to incorporate all these actions into a single model. Therefore, systems require multiple models that provide different functionality, which the system subsequently combines in various patterns to provide the full functionality.

We recently published the *Reference Architecture for Assuring Ethical Conduct in LAWS* as a companion document to this guidebook [Mellinger 2025]. In the reference architecture, we provide an example of a mobile autonomous system with a fully autonomous perception, targeting, and response (PTR) system that also demonstrates an approach to ethical conduct governance. While we do not expect developers will necessarily reproduce the organization or the exact components of the architecture as we describe them, we do expect that system development and resulting systems should focus on the same quality attributes of reliability, robustness, safety, and usability that we do in the reference architecture.

We produced the reference architecture to reason about the challenges we described in Chapter 3 and to provide examples of how any system should be able to answer the following questions:

- What models are used by the system and what are their inputs and outputs?
- How are inputs to the models validated and how is that validation used to support decision-making, how is decision-making communicated, and how is it stored for later analysis?
- How are model outputs checked, and if necessary, constrained? How is that information communicated to the operators and stored for later analysis?
- What are the system's decision-making components, and what is their basic decision-making logic? How do they deal with questionable input data or model outputs?
- How is the decision-making process communicated to the operator?
- Which components are responsible for the application of ethical controls, how do they apply these controls, how do they communicate with the operator, and how do they store their information for later analysis?
- How are ethical guidelines entered into the systems and configured for regions, missions, or commander's intent?

The answers to these types of questions should be evident in the system-level architecture to allow appropriate construction of system testing during developmental and operational testing. For example, based on input images, a test should be able to show which components are responsible for identifying mission-relevant features from the sensors, which others are making these decisions, which provide ethical controls and thresholds, and which logs show this output.

Many of the questions above ask how the system communicates the various types of information to the user. To support such communication, the system design should leverage pervasive architectural affordances with corresponding user-interface support and robust logging infrastructure. The "IEEE Standard for Transparency of Autonomous Systems" provides a comprehensive guide to transparency including how to perform a system transparency assessment, how to write a system transparency specification, and how to prepare a transparency design process [IEEE 7001-2021].

5.1 Commentary on ML System Design Testing

Each component of a system has a different set of requirements, and these requirements are often not qualitatively similar. Earlier, we made the distinction between ML and AI where ML refers to the model and AI refers to a combination of the model and additional symbolic code to interpret the predictions for use in decision-making. The requirements for ML components usually contain details such as precision and recall, and they require tabulating component behavior over a large amount of data. On the other hand, requirements for software components such as AI typically describe behaviors or input and output pairs. TEVV personnel usually test these software components using unit tests, where the function is called, perhaps a dozen times, to exercise all the code paths and check that the component behaves as expected. This type of testing is completely different from testing that runs thousands or more images through an AI and ML component to understand general trends in its performance, and, therefore, such testing requires different tools.

System requirements can either describe explicit behaviors (e.g., with a statement like “The robot reaches its goal location.”), or they can be performance based (e.g., with a statement like “The robot can traverse this course in less than 10 minutes.”). Tests for these requirements must go beyond simple functionality checks that TEVV personnel typically perform for software components in isolation, and they require a different testing approach to accomplish that goal. The following list offers a rough guide for writing requirements:

- ML components will require evaluation over a dataset to compute values like precision and recall.
- AI and ML subsystems (e.g., an object detector and tracker pair) may require a more advanced version of testing that occurs over a dataset to compute values.
- AI software components will require unit tests that show that the component performs all the expected behaviors correctly.
- Hardware, mechanical, and electrical components will often require physical testing as well as analysis in software.
- Overall system testing requires either field testing, simulation testing, or both, and it can also make use of log-replay based testing
- TEVV personnel can also use log-replay based testing on all types of subsystems.
- Some components may need (or at least may benefit from) different testing approaches for different requirements.

These are rough guidelines and are not meant to be prescriptive. However, every single requirement must have at least one test that checks it. These approaches are common as they are often the simplest, most cost-effective, and most repeatable way of performing that testing.

6 Model Design, Development, and Testing

This chapter provides guidance to assist development teams and TEVV personnel with the selection, design, and training of machine learning (ML) models used in lethal autonomous weapons systems (LAWS). Importantly, there is a strong foundation for performing development and testing of individual models, with scores of supporting research and documentation. We strongly urge TEVV personnel to become acquainted with those available resources. However, one of the goals of this guidebook is to recognize that existing sources are often insufficient to address the special characteristics and needs of LAWS. This chapter, therefore, provides guidance for navigating useful resources, but it also aims to build on those resources to establish actionable guidelines for meeting the unique conditions of LAWS.

Among the numerous sources that exist to help development teams establish effective practices for developing and testing ML models, we recommend that they review *Machine Learning in Production* by Kästner, *Artificial Intelligence: A Systems Approach from Architecture Principles to Deployment* by Martinez, and the DIU Guidelines and Worksheet for the RAI perspective [Kästner 2025; Martinez 2024; Dunnmon 2021]. In addition, full-featured development platforms and pipelines are readily available for ML development, and developers and TEVV personnel should expect to gain modern engineering benefits from these practices such as configuration management, continuous integration, automated testing, version control, and so forth. However, as we explain above, this chapter does not cover these details but instead focuses on how LAWS differ from other systems and the needs these differences occasion.

Although there are many different ML development process models, we have chosen to concentrate on a particular version of machine learning operations (MLOps) for this guidebook, which is depicted in Figure 44. MLOps assumes the model is placed directly into operation (hence the "operations" portion of the name), but for our purposes, providing it for system integration is equivalent to deploying it into operations, so the model is representative of what LAWS need. Figure 44 shows an example MLOps process that we have chosen because the process explicitly highlights the RAI principles and emphasizes the connections and interactions of the process with those principles. Application of RAI and other ethical controls in LAWS is crucial because the consequences of failure are so high.

Figure 4 comes from an SEI whitepaper on counter AI called *Counter AI: What Is It and What Can You Do About It?* which we highly recommend as background reading to understand the basic approaches attackers use to launch attacks on AI, and what development teams can do to prepare for such tactics [VanHoudnos 2024]. In addition, see Chapter 2 of this guidebook for discussion of the ethical motivations and principles we outline for LAWS.

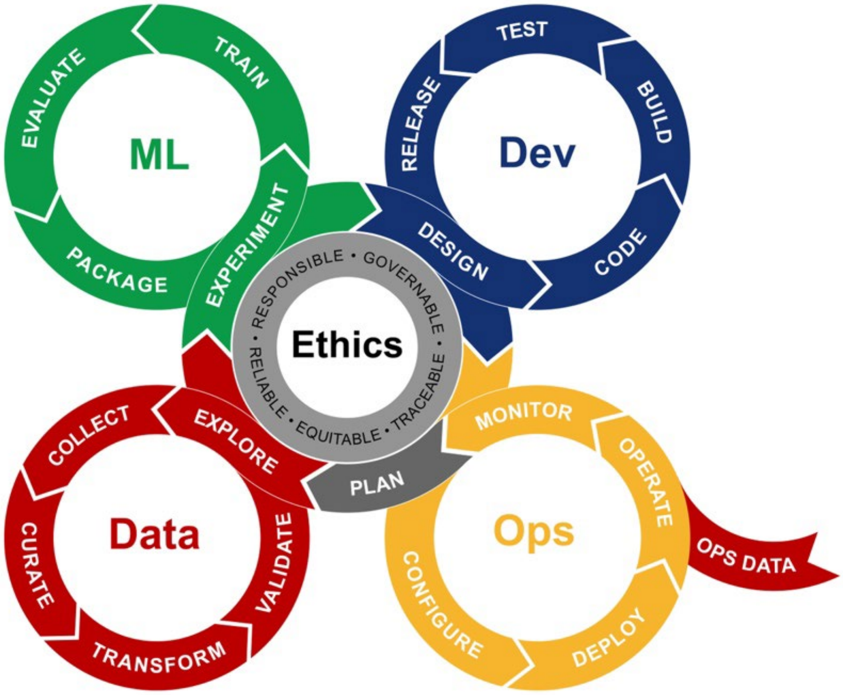

Figure 44: MLOps Model

Using Figure 4, development teams can begin thinking through the process of choosing a model architecture and establishing a process for model development that addresses the most pressing issues of developing ML for LAWS. To start, models function by establishing an *architecture* that describes the types of system layers and the relationships between them—sometimes organized into groups—that provide the model with its distinctive style, such as a convolutional neural network. There are many different model architectures to choose from that provide different layer group types and repetitions of groups, as well as different group sizes, customizations, and even pretraining options that development teams must choose before they begin the iterative development process. Model architectures have a direct impact on traditional model performance measures such as accuracy, precision, and recall, but also on other considerations such as memory utilization, runtime performance, and training cost. Choosing an inappropriate architecture can lead to significant rework. We cover model selection in Section 6.2.1.

After base model selection, developers must train or tune models on mission specific data, which sometimes requires model modification (e.g. changing layers) beforehand. We discuss how to adjust the data during training and other training pipeline options in Section 6.2.3 and Section 6.2.5. In Section 6.2.4, we discuss how to select metrics for training and how to know when basic training is sufficient. We discuss how to prepare the model for counter AI in Section 6.2.9. Finally, we discuss how to prepare the model for handoff with appropriate artifacts in Section 6.2.8. To learn more about how to calibrate the model for mission use, reliability, and robustness, readers can refer to Heim's *A Guide to Failure in Machine Learning: Reliability and Robustness from Foundations to Practice* [Heim 2025].

6.1 Recommendations for Model Design, Development, and Testing

The following sections provide a list of recommendations for development teams and TEVV personnel to appropriately and effectively design, develop, and test the system.

6.1.1 Start with Good, Appropriate Models and Metrics

Model architectures and implementations can have tremendous impacts on the quality of the model outputs, but TEVV personnel must measure them appropriately with respect to the mission. We discuss model architecture in Section 6.2.2 and the selection of appropriate metrics in Section 6.2.4.

6.1.2 Tailor the Pipeline to the Mission Need

Having good data and a suitable architecture isn't enough to ensure high-quality output from the model. To ensure the highest quality output as possible, the training pipeline and test infrastructure must follow best practices and must match the model and mission needs. We discuss training practices in Section 6.2.3, responsible engineering in Section 6.2.7, and metrics selection in Section 6.2.4.

6.1.3 Get Control of Probabilistic Software Development Experiments

Building successful, complex probabilistic systems such as ML requires that development and TEVV teams develop a strong experimental mindset and acquire tool support. Controlling random seeds and sources of non-deterministic behavior become necessary for diagnostic and comparative analysis. We discuss this topic further in Section 6.2.1. Tests must accommodate these controls, which we discuss in Section 6.2.6. This control is part of the overall responsible engineering practices, which we discuss in Section 6.2.7. Finally, we discuss the documentation of these configurations in Section. 6.2.8.

6.1.4 Confirm Through Testing That Processes Are Effective

Good data, architectures, designs, and processes promote good outcomes, but they must be confirmed through testing as well as any feature. For example, development teams might have made use of a good model selection process, but TEVV personnel must still test the model after selection, and it should be compared against alternatives. We address how to design good model tests in Section 6.2.6, how to control the probabilistic nature of these tests in Section 6.2.1, and how to document them in Section 6.2.1.

6.1.5 Consider Robustness and Model Calibration When Designing the System

Accuracy, precision, recall, and other traditional model metrics are important, but there is much more to modern model testing such as robustness to adversarial attack and model calibration. TEVV personnel should consider and address these techniques during training and make sure that the model's outputs provide evidence to support that development teams have properly prepared the models. We discuss counter AI in Section 6.2.9 and outputs in Section 6.2.8.

6.1.6 Segment Testing for Performance Insights

TEVV personnel should perform a detailed breakdown system performance by segmenting the test set to reflect important properties or circumstances for which TEVV personnel need details. TEVV personnel should then filter results by time of day, location or environment, or classes of interest. All of these can provide more detailed information about the system's strengths and weaknesses.

6.2 Commentary on Model Design, Development, and Testing

The sections that follow provide more information about the recommendations in Section 6.1.

6.2.1 Repeatability, Reproducibility, and Replicability

Reproducibility is part of any mature testing process. Performing this part of testing is more challenging and important for MLES than for other systems, but TEVV personnel must plan and account for it. Development teams should allocate more resources and planning time than they normally would in traditional projects to provide reasonable levels of reproducibility. However, as model training can result in very large outputs, saving every output at every stage is intractable due to the amount of storage required. Conversely, attempting to regenerate intermediate outputs dynamically, such as re-running earlier experiments, may be too costly in terms of time and computing resources. Development and testing teams should reach early agreement about the appropriate amount of infrastructure and tooling required for anticipated reproducibility needs.

Additionally, establishing the repeatability, reproducibility, and replicability of testing is important to support diagnostics and experimentation as well as the overall assurance case.

Reproducibility as a general term covers a wide variety of situations. To help establish what we mean by the terms repeatability, reproducibility, and replicability, we leverage the definitions offered by the Association for Computing Machinery (ACM), who offers useful distinctions for these three terms. ACM defines the terms as follows:

> *Repeatability (Same team, same experimental setup)*
>
> *The measurement can be obtained with stated precision by the same team using the same measurement procedure, the same measuring system, under the same operating conditions, in the same location on multiple trials. For computational experiments, this means that a researcher can reliably repeat her own computation.*
>
> *Reproducibility (Different team, same experimental setup)*
>
> *The measurement can be obtained with stated precision by a different team using the same measurement procedure, the same measuring system, under the same operating conditions, in the same or a different location on multiple trials. For computational experiments, this means that an independent group can obtain the same result using the author's own artifacts.*
>
> *Replicability (Different team, different experimental setup)*

> *The measurement can be obtained with stated precision by a different team, a different measuring system, in a different location on multiple trials. For computational experiments, this means that an independent group can obtain the same result using artifacts which they develop completely independently [ACM 2020].*

For model training, these definitions have several implications that TEVV personnel must consider. For example, when a team is working together to train a model, repeatability enables the team to make incremental changes to code, to hyperparameters, or to other properties and to determine the impacts of those changes. Teams do not need to complete all the training steps at one time, but they may instead perform iterative model training because each subsequent epoch or iteration of training can be considered as tuning the model. However, supporting a process where developers repeatedly tune a partially trained model requires re-initializing the training system to the last state, which in turn implies strong reproducibility. Based on the above three definitions of reproducibility, it is therefore important to define the conditions in which the iterations take place. For example, TEVV personnel should specify whether all tuning iterations happen on precisely the same training system (hardware and software), or whether the iterations can happen across different systems and at different times.

Replicability requires a significantly higher level of configuration, capture, and control, and it may not be necessary for the development of the models and systems. A fully replicable process would require identical hardware, identical software platforms, identical starting conditions, identical software processes, and full control of all random seeds, networking control, and process interactions. Achieving replicability for a fully formed system is extremely expensive and probably isn't worth the value it provides.

Reproducibility is difficult to build into the system after development has started. Therefore, development and testing engineering teams should work together at the beginning of the development process to agree on the level of reproducibly required at the model level. They also need to discuss the implications these requirements will have on training time and diagnostics.

Experimental control and reproducibility introduce additional challenges when stochastic components are involved. ML training intentionally introduces randomness for many reasons, such as preventing overfitting, and it allows a level of non-determinism to occur to allow optimizations for runtime training performance [Mellinger 2025].

6.2.2 Base Model Selection

This section discusses the many factors that developers should consider when choosing a model to fit the system's purpose. We provide overviews of the basic properties of models that developers should consider when assessing whether the model is appropriate for the task it is intended to perform. These considerations should occur before beginning the test and evaluation (T&E) process.

ML models can serve different purposes such as image processing or language tasks. For LAWS, we expect that designers and developers will choose models for image processing tasks such as image classification, object detection, or object tracking. For more information, we provide discussion in Chapter 5 of different design patterns that developers can use to form a perception, targeting, and response chain. In that chapter, we also discuss these behaviors, and we offer examples of several models that developers can use in that pattern.

Model developers should describe the architecture and implementation they chose and why, and TEVV personnel should review these descriptions as they are necessary for developing testing. For example, the You Only Look Once (YOLO) object-detection system is a popular, single-shot object detector architecture that different organizations have used in many implementations for numerous purposes. YOLO models are not all produced by the same group or vendor. In fact, YOLOv5 and YOLOv7, while based on the YOLO design, are completely different versions made by completely different vendors. Unlike traditional software, where a major version change implies significant changes to features, in this case it may also involve a complete change of developer, implementation, and properties, which increase the workload for TEVV in that it violates a lot of common assumptions about software. For example, YOLOv4 or Tiny-YOLO, are versions of YOLO intended for small computing formats such as a Raspberry Pi. Roboflow's "What is YOLO? The Ultimate Guide" is an excellent resource that explains the history of YOLO models and even provides visualizations to compare its different versions should TEVV users need to understand the differences between different versions of YOLO [Roboflow 2025].

After system designers and developers choose a model architecture, they must decide whether to use a pretrained model. Many models are pretrained on ImageNet even though that resource doesn't provide the dataset or classes that the system will need to function for the purposes of most LAWS. Models pretrained on ImageNet provide a good set of starting weights instead of a random distribution [ImageNet 2025]. Many organizations, such as Nvidia, provide models that they have tuned for special purposes such as just identifying people [Nvidia 2024]. There are other publicly available models that are tuned to detect street signs or cars, and developers can tune any of these models even further. However, it is important for TEVV personnel to understand how the model was trained. Understanding the data that developers used in all stages of the training process can inform testing about particularities in results.

Tuning a model often involves replacing the last layer in the model with a new one that developers fine-tuned for new classes and images. Alternately, it may just involve freezing some sets of layers and replacing or training others, while other layers may be added at the end for other purposes such as calibration. Developers should note and document any changes to a base model architecture, and TEVV personnel should ask for the reasoning behind the changes and descriptions of the implications and side effects of the changes. TEVV personnel should test the model for any side effects or performance impacts not anticipated during the design process.

Lastly, developers might choose certain models because of runtime performance. Some models, such as TinyYOLO, are optimized for small memory or low processor usage, which often negatively impacts model performance such as accuracy, precision, or recall [Rosebrock 2020]. While this result is often a necessary tradeoff, these impacts should occur by design, and developers should document them. Naturally, these changes negatively impact model performance, and performance metrics on these smaller model versions will be lower than those on fully featured models, so normal benchmarks should not be used for comparison. For example, a system using TinyYOLO will not be able to achieve the same model performance numbers as their full-sized versions. Models working on small devices can also suffer from other real-world impacts because they operate on lower resolution sensor images, which limit vision range. Developers must identify and document all side effects, and TEVV personnel should construct appropriate tests to make sure models meet mission goals.

6.2.3 Training Practices

Pretrained models are unlikely to be appropriate for a LAWS. One issue is that few datasets for pretrained models will contain information about military equipment, and those that do will most likely not contain the level of detail required for appropriate training. For example, we expect that datasets will need to use mission-specific class distinctions, such as those that distinguish between military and civilian vehicles, and they will need to use classes for objects in unusual states such as vehicles that show battle damage. Accurate perception of such differences requires some level of training or tuning in a perception chain.

To train or tune a model, developers repeatedly feed training and validation data into the model until it reaches appropriate metrics. In Section 6.2.4, we discuss how to choose appropriate mission metrics to perform this process.

There are many techniques that developers can use to organize and feed the data into the model during training that can influence the outcomes such as overfitting. However, all these techniques will impact training time. If the final measure of a good model is how it performs, then the process of getting to a final model is not important except the time and cost. However, ML deployment differs from traditional systems deployment in that we expect the models to need updating (retraining or tuning) over time. Therefore, delivering a process to update the model is part of the output. T&E should choose a functional model, but it must also choose a model based on how feasible it will be to establish a process and executable pipeline that allows quick, efficient, and cost-effective updating without sacrificing mission quality. Existing T&E capability may not be prepared for evaluating these processes and their impact on mission quality as part of a product deliverable, so development teams should allocate additional time and resources to ensure that testing is adequate for these outputs.

Currently, there are no standardized guidelines or benchmarks for establishing a retraining or tuning pipeline that meets mission purposes, but resources such as *Machine Learning in Production: From Models to Products* provide a good starting point for evaluating such pipelines [Kästner 2025]. Any such evaluation should cover the ingestion of new data including data preparation; adjustments to the training process (such as number of epochs or iterations); quality metrics such as accuracy, precision, or recall; target classes; pipeline augmentations for robustness; and frameworks for counter-AI checking, to name a few.

Training might sometimes entail introducing more complicated changes, such as the addition of new target classes as part of a change to mission profile. Such changes may result in significantly new mission capabilities, and they might require senior review as per DODD 3000.09, Section 1.2.c.2, which states that senior review is required for changes to any of the following: “system algorithms, intended mission sets, intended operational environments, intended target sets, or expected adversarial countermeasures [that] substantially differ from those applicable to the previously approved weapon systems” [DODD 3000.09]. Data classes most likely will change “intended target sets” and updated counter-AI libraries can significantly change “adversarial countermeasures” as well. In short, any sort of update or change to the model or pipeline during updating, even without significant change to the deployment code base, can significantly change the behavior from a DODD 3000.09 perspective and must be re-approved. This level of change will be inevitable. Pipelines, therefore, should be built in a way that developers can easily introduce changes, and TEVV personnel should prepare a T&E process that is at the ready to evaluate these changes. To extend this scenario further, TEVV personnel should prepare a T&E process that can readily evaluate and consider any changes that might occur in the MLOps model described in Figure 44. See section 6.2.6 for a further discussion on pipelines.

During training, it is important to understand if the training process affects the model, what sort of an effect it is having, and how much. There are a variety of diagnostic utilities that developers can apply during training, such as during backpropagation, to evaluate changes to the model, to understand the effectiveness of the training process, and to gauge the impact of any changes to the process. An example of such a utility are saliency maps, which developers can use to identify what part of the images are being identified by which layers. Developers can also use other activation-based tools to identify which layers are being affected by changes to inputs such as augmentations for robustness or for the prevention of overfitting. Describing the details of these activities is beyond the scope of this guidebook, but evidence of their use and effectiveness should be identified by T&E as part of the assurance case.

6.2.4 Metric Selection

Training typically involves iterations in which developers train the system using batches of training data and then measure system performance using batches of validation data. This process is repeated until the model meets a given quality metric for the testing predictions it makes about the validation batch. A common metric is *accuracy*, which refers to the basic measure of how often the model’s predictions are right. Other metrics provide more nuanced measures such as *precision* or *recall*, and still other metrics combine these measurements into new metrics, such as the *F1 score.* The Organisation for Economic Co-operation and Development (OECD) maintains a list of 130 metrics in their “Catalogue of Tools & Metrics for Trustworthy AI” [OECD 2025]. Developers and TEVV personnel must select the metrics that best measure the performance of the system to ensure it meets mission goals.

Along with a wide variety of mathematical metrics, the T&E process must determine the right mission value for each metric and whether testing should apply different values for different system contexts or outputs. For example, developers and TEVV personnel must consider whether to apply the accuracy metric against all classes evenly, or whether they should apply different thresholds per class. This is an important consideration because it’s possible that a particular use case might require that the system identify civilians with greater accuracy than domestic animals. Alternately, developers and TEVV personnel might apply a different threshold for *recall* (i.e., how well a model can identify positive instances in a dataset) to different classes.

Given the complexity of mission scenarios, we expect that TEVV personnel will measure a model’s fitness for purpose using values that vary across multiple metrics and many different classes rather than using a single value or metric such as accuracy. In addition, it is reasonable to expect different thresholds and measures based on use case or context variations such as environment, lighting, activity, and weather.

6.2.5 Overfitting

When training an ML model, it is possible to *overfit* the model to the data, which means that the model is trained very specifically to that dataset and does not perform well with regards to real-world data. A system yielding high accuracy rates (or other metrics) on the training data but poor values on actual test data can indicate that overfitting has occurred.

The basic strategies to prevent overfitting involve acquiring a larger data set, permutating the source data so that it isn't the same during training iterations, creating synthetic data if possible, or discontinuing training when the system meets a certain metric. Augmentations for image data include techniques such as rotations, scaling, and cropping, which can make the original source data appear differently without changing the data. Developers can also introduce true modifications to the source data, such as modifying light levels, adding artifacts, or adding synthetic data, but doing so can introduce certain risks, which we discussed in Chapter 4.

6.2.6 Designing Model Tests

There are many resources and approaches that provide good guidance for model testing, and it is up to each program to find one that fits their culture, process, product, and mission goals. The approach that programs choose to adopt must address the entire lifecycle and integrate with requirements and the development process in a meaningful way and generate verification and validation (V&V) testing commensurate with the mission.

TEVV personnel can consult the following resources and combine them to assemble a comprehensive approach:

- Chapters 14 through 18 of Kästner's book, *Machine Learning in Production: From Models to Products*, provide a good resource to plan for data management, model testing, pipeline management, and system quality [Kästner 2025].
- "Test & Evaluation Best Practices for Machine Learning-Enabled Systems" provides excellent coverage of the challenges and approaches to generalized ML model testing. It covers areas such as test generation, test adequacy, and integration and deployment. Test generation addresses how to reapply common traditional testing techniques such as metamorphic testing, differential testing, combinatorial testing, and fuzz testing to ML. It also provides some discussion of adversarial testing [Chandrasekaran 2023].
- The *MLTE Documentation* website developed the Machine Learning Test and Evaluation (MLTE) project, which contains a testing library and process framework that starts with mission analysis facilitated by quality attribute scenarios [MLTE 2025]. It also introduces the idea of *negotiation cards* (like data and model cards, but for testing based on quality attributes), discusses how to apply them to various stages of model testing, promotes iterative processes, and provides a test catalog and guidelines for producing reports. These techniques are further discussed in the

papers “Using Quality Attribute Scenarios for ML Model Test Case Generation” and “MLTEing models: Negotiating, Evaluating, and Documenting Model and System Qualities” [Bower-Sinning 2024; Maffey 2023].

- The Chief Digital and Artificial Intelligence Office’s (CDAO’s) Joint AI Test Infrastructure Capability (JATIC) toolkit contains a wide variety of ML model testing and evaluation tools that address data quality management, data augmentation, labeling, visualization, and counter AI robustness [CDAO 2025].
- ISO/IEC 5338 section 6.4.11.3 points out that TEVV personnel should perform verification of AI based systems from a behavioral perspective [ISO/IEC 5338:2023].

The resources above are not specifically geared for LAWS. Therefore, TEVV personnel will need to adjust testing plans for the specific safety, ethical, and HSI concerns that we described in Chapter 2 and Chapter 3 in addition to the safety concerns addressed in MIL-STD-882E.

TEVV personnel should construct test variations and metrics in accordance with the readiness of the model and the system in which it will be injected. ML technology readiness levels (MLTRLs) can provide guidance to understand data maturity and review at each stage of the model and system testing [Lavin 2022]. These processes become more important at the system level, and we discuss them in depth in Chapter 8.

6.2.7 Responsible Engineering Practices, Processes, and Tools

As of this writing, there are few fully featured and mature tool chains for the TEVV of MLES, and none specifically focus on trust and trustworthiness. We want to stress for the purposes of this guidebook that—although numerous documents exist that attempt to describe how to implement RAI, how to add ethical processes to systems, and how to define RAI maturity models [IEEE 7000-2021; DOD 2022a]—many of them are unlikely to extend seamlessly to the needs of LAWS. RAI tools and practices are co-evolving but few, if any, industrial tools have incorporated the principles and tenets of RAI at the level needed by LAWS. Some standards explicitly exclude ethics. Such is the case with UL 4600, which states, in Section 1.3.4, that “Two areas out of scope for this standard are setting acceptable risk levels and setting forth requirements for ethical product release decisions and any ethical aspects of product behavior” [UL 4600]. While understandable for a product that fits into an already regulated environment, the implication is that any guidance in this standard may not be directly applicable to LAWS.

Until there is an “RAI approved” or “RAI compliant” tool set, each program will have to make its own. The burden therefore falls to the developers and, as usual, the TEVV community to not only examine the products themselves, but also the processes for integrating these products. The Defense Innovation Unit (DIU) guidelines and worksheets for RAI and the Institute of Electrical and Electronics Engineers’ (IEEE) ethical process guidance can both serve as useful references during ethical model development [Dunnmon 2021]. Efforts such as JATIC are currently trying to assemble modern tool suites, but few offerings are established, much less commercial. The operating assumption of TEVV personnel should be that most tools, practices, and tools chains have not been designed to conform with or support RAI practices. Software engineering has many guides and techniques for assessing and selecting tools, and programs should consider using a process such as the one suggested in "A Process for COTS Software Product Evaluation" [Comella-Dorda 2004]. Lastly, teams should consult conceptual frameworks, like "Canonical Stack of Machine Learning," that elaborate on possible components for a complete tool chain [Jeffries 2022].

6.2.8 Model Cards, Outputs, and Standardization

The process of training creates not only a trained model but also generates useful information about the model such as accuracy and loss, how it performs against each test set, and other useful information about the training process and the training pipeline. In this section, we discuss some techniques that developers and TEVV personnel can use during the training process to capitalize on opportunities beyond just training a model, such as capturing key information.

Model cards—which are like the data cards we discussed in Chapter 4—were initially proposed in 2018 in a research paper by Google research to provide a semi-formalized way to capture the a technical description of the model including the model architecture, its hyperparameters, its training process, and so forth [Mitchell 2018]. Since then, many organizations have published model cards and improved what they capture by adding more fields or requiring formats such as YAML. A popular data and model-sharing website called Hugging Face further refines this idea with additional definitions and provides a template [Hugging Face 2025]. The model cards at Hugging Face are more formalized and machine readable than other, commonly available formats, but they still contain a lot of information in natural language, which presents challenges when trying to analyze and summarize a system containing many models. As for data cards, efforts are underway to standardize even more of the terminology and formats to improve machine readability.

In Section 6.2.6, we discuss how to create model tests and how to capture testing goals and outputs using standardized processes and formats that promote completeness, interoperability, and reuse. TEVV personnel can find examples of testing cards—which bear similarities to data and model cards—in MLTE's negotiation cards and in the Systems Engineering Processes to Test AI Right (SEPTAR) framework's evaluation cards. Although the cards from MLTE and SEPTAR serve different purposes, they both provide common ground and techniques to understand user needs and model outputs and to how TEVV personnel can produce and organize test outputs. As with other approaches, there is room for additional formalisms and standardization, but they provide a good start to create a full-featured suite of cards for describing the data, models, and performance.

How developers build a model is just as important as the model itself, and it is important to remember that the entire collection and logs that comprise the training system will contribute to the overall assurance case. In the article "Traceability for Trustworthy AI: A Review of Models and Tools," the authors propose a single, unified file format that describes the business, data, modeling, processing pipeline, evaluation criteria, and deployment that serve as an example of a complete set of output data with formalisms [Mora-Cantallops 2021]. Until there is a standard for what comprises a complete AI assurance case—such as something similar to one listed above—each program should define guidelines and details for each technical area such as the data used for training, the model development process, model evaluation, or pipeline development. Further, programs should consistently use these templates across all efforts of ML model development. The final data-model-pipeline-evaluation chain is a central piece of the assurance case to support the RAI principle of traceability.

Each model and mission are different, and developers and TEVV personnel will need to define different testing thresholds to make sure ML models meet their specific mission goals before developers integrate the models into a system. However, TEVV personnel can and should reuse testing tools, types, and strategies between models as much as possible for consistency. There are many examples of model tests available that TEVV personnel can use and repeat for these purposes, such as those in MLTE's test catalog or OECD's metric's use cases and [Bower-Sinning 2024; OECD 2025]. TEVV personnel should give preference to these existing tests over novel approaches when appropriate.

Additionally, in systems with multiple integrated models— which we described in detail in Chapter 5—testing should be complementary across all models. TEVV personnel should choose measures that reflect consistent mission goals, while tailoring thresholds per model in the system when appropriate. However, TEVV personnel must identify differences in model goals such as static classification as opposed to motion tracking, and they should test for these different goals separately while keeping in mind how they contribute to overall system goals. To support these tasks, we discuss system testing in detail in Chapter 8.

The techniques and approaches are intended to apply to general types of AI, not just LAWS. There is currently no clear guidance on which approaches are most useful for LAWS in general, or for LAWS in specific mission domains or use cases. Therefore, each program will need to identify in the early stages of development which metrics are most appropriate to the mission at a system level and then derive associated model-relevant metrics.

6.2.9 Counter AI

In their whitepaper, *Counter AI: What Is It and What Can You Do About It?* the authors provide an overview of where counter AI attacks occur in the technology stack, and they identify three attack types—*learn the wrong thing, do the wrong thing,* and *reveal the wrong thing*—along with five different threat models [VanHoudnos 2024]. Many of the defenses against these adversarial threats will occur at the system level through validation checks on sensors, preprocessing steps to mitigate errors, and hybrid AI and ML, multi-component architectural approaches to mitigate erroneous outputs. We discuss these approaches in more detail in Chapter 5 and Chapter 8.

Testing efforts must account for adversarial attacks during evaluation, regardless of evolving terminology. ISO/IEC 25059:2023 defines robustness as "the presence of unseen, biased, adversarial or invalid data inputs" [ISO/IED 25059:2023]. By explicitly including "adversarial" in this definition, the standard now directly incorporates counter AI and expands robustness to include defenses against adversarial machine learning operations. Previously, it was unclear whether such testing should be conducted as part of security assessments, whether existing reliability or robustness measures were sufficient or inclusive, or whether these considerations were covered under existing areas at all. Regardless, they need to be accounted for at all levels of testing.

Counter AI is an active area of research, and researchers are working on many emerging tools and approaches to address it. For example, the JATIC tool set includes the Adversarial Robustness Toolbox (ART), the Hardened Extension of the Adversarial Robustness Toolbox (HEART), and the DARPA Guaranteeing AI Robustness Against Deception (GARD) program's Armory platform [CDAO 2025; Two Six Technologies 2025]. These available tools can prove useful for developers and TEVV personnel to safeguard the ML models in operation in the LAWS they are developing and testing. Developers and TEVV personnel could also benefit from staying informed about the release of any new and emerging tools or approaches that researchers are currently developing.

From a model perspective, TEVV should look find evidence that training and test datasets prepare the system to manage adversarial information such as label attacks, adversarial patches, or other misleading information such as the attack types we mentioned above with respect to the counter AI whitepaper. The MLOps pipeline should show evidence that developers have deployed tools to bolster robustness—especially with respect to including defenses against counter AI—and that test results include metrics that show the effectiveness of the tools and testing included in the system to combat counter AI.

7 System Development

In this chapter, we discuss how developers integrate models into the system during development, and we address the kinds of challenges that the development of machine learning (ML) and artificial intelligence (AI) for lethal autonomous weapons systems (LAWS) specifically introduces. This chapter and Chapter 5 work together to bookend the model development portion of this guidebook. In Chapter 3, we discussed concerns about developing LAWS with ML and much of our commentary addressed how to properly specify and design the systems. As we've noted in other parts of this guidebook, we do not intend to duplicate existing work. Rather, our intention is to provide guidance for integrating ML systems into LAWS.

Chapter 5 is important for this chapter in two ways: First, the suggested reading we outline in that chapter applies to our discussions below, and readers should keep those readings top of mind throughout this discussion. Second, readers should refer to Figure 44 for an overview of our understanding of ML operations (MLOps) as they reflect on the points we make below.

7.1 Commentary on System Development

In the sections that follow, we highlight specific differences that teams will encounter when developing ML systems. Understanding these differences will help testing, evaluation, verification, and validation (TEVV) personnel to identify potential process gaps that may need additional effort. The following sections outline the main differences we want to highlight to help with that process.

7.1.1 Allocate Extra Time for Process and Technology Maturation

ML is still in early development, and the community is rapidly iterating on techniques and technology. Many tools prioritize getting emerging techniques into the hands of researchers over production use. Developers that intend to use the latest technology for LAWS will likely end up using experimental code. Developers should prepare for common maturity challenges such as changing codebase, defects, minimal diagnostic and debugging support, and incomplete or inaccurate documentation.

7.1.2 Prepare for Higher Computing Resource Demands

ML models can be very demanding of memory and processor resources. Therefore, developers must design simulations, development, testing tools, and infrastructure to accommodate for this issue. There are a variety of efforts to develop smaller and more efficient models, more efficient execution engines, and even custom hardware designed for specific models to relieve the strain that ML models put on memory and processors. These projects are under intense development, and each one incurs its own challenges and impacts on development timelines for ML models.

7.1.3 Build One to Throw Away (You Will Anyway)

The common software development adage—build one to throw away—holds true for the LAWS development process across many dimensions including models, system constructs, pipelines, and test infrastructure. While we often don't actually throw these systems away when using iterative and refactoring process, we must be prepared to make significant changes from the original design during the development process. ML systems are particularly prone to needing such adjustments due to how fast the operational environments in which models are deployed can change.

7.1.4 Team Culture May Not Support Testing

Testability is commonly taught as part of the architecture and design portions of the software engineering curriculum [Clements 2011]. The community of engineers and practitioners working on LAWS evolved out of the autonomy community and ML communities. Because the LAWS community is often not rooted in software engineering practice, it may suffer cultural issues such as roboticists not wanting to do testing it which is discussed in detail in the paper "A Study on Challenges of Testing Robotic Systems" [Afzal 2020]. When beginning the development process, it is important for development teams to identify organizations' and teams' existing challenges to determine what changes to the development process and team culture are necessary to accommodate the additional demands of ethical development, ethical systems design, and the RAI principles.

7.1.5 Allocate More Time and Effort to Develop and Manage Baseline Tests

Developing baseline tests for autonomous or other non-deterministic systems must account for that variability, which requires extra planning, effort, and resources. Setting a priority to perform early and regular experiments and to establish baseline development (one of the tenets of Agile, to deliver early and often, is relevant) can deliver tremendous rewards. TEVV personnel should focus experiments on sets of well-understood and controlled changes so that they can compare the output from subsequent experiments to the previous results. This approach applies to experimentation with models, pipelines, and data changes. The more precise the experimental set, the better. Obviously, this approach work will require more training time, but by using smaller, well-crafted sample sets, shorter training times, and the payback from easier debugging, the experiments will quickly pay off. See Chapter 4 for details about how to make appropriate sampled sets.

7.1.6 ML Development Tools Have Higher Resource Demands

Configuration management of ML-enabled systems (MLES) as opposed to traditional systems requires significantly more resources to deal with data and model versions. Traditional version-control systems designed for code are not well-equipped to handle the kind of versioning needed for data and models due to the large resource demands. In addition, any storage systems must support metadata tracking for the purpose of establishing provenance. Development teams often overlook or underestimate these needs during project inception because many experienced teams have already created a well-established environment and are unprepared for the changes that new ML components introduce to their processes.

7.1.7 Placeholder Components Provide Extra Value During MLES Development

Developers can proceed with system development while model development is still underway by using placeholder models or *mocks.* Mocks are well-controlled, test infrastructures with the same API as the model under development, and developers and TEVV personnel can use them while models are unavailable because they enable system testing by returning regular and controlled values. While mocks are useful in normal practice, they are more useful in systems that use probabilistic models. During development, developers should try to retain the ability to independently configure which models are mocked so they can diagnose issues and conduct other controlled experiments.

7.1.8 Code Instrumentation Provides Higher ROI for ML Systems

Developers should fully instrument the boundaries of each model. Development code should have even more instrumentation than validation and checking concepts of runtime code so that developers can monitor the models, validators, and checkers during development. Developers should pay special attention to ensure that instrumenting the connection from model to decision-making allows full traceability and governability analysis for transparency. See Chapter 3 and Chapter 5 for details.

8 System DT&E

System developmental testing and evaluation (DT&E) focuses on the testing and evaluation of a system's design and performance during development before it proceeds to more extensive testing and operational deployment. DT&E aims to identify potential design and functional issues early in the development process, ensuring that machine learning (ML) systems meet stringent operational, security, and ethical standards before full-scale implementation.

We expect that readers of this guidebook will have a good understanding of systems development and systems testing, so we don't focus on the foundational details of those processes. Rather, we focus on the special considerations that ML brings to the development and testing of LAWS and how these introduce important differences to generally established approaches. This chapter, therefore, focuses on what those considerations and differences are for DT&E.

Studies that provide significant general knowledge on the development and testing of ML- and artificial intelligence- (AI)-enabled systems is forthcoming, and our goal is to not replicate those efforts [Huyen 2022; Bass 2025; Kästner 2025].

Testing, evaluation, verification, and validation (TEVV) personnel can benefit from consulting several resources that focus on how to adapt existing processes to the special considerations introduced by AI. One such resource is ISO/IEC 5338, titled "Information technology—Artificial intelligence —AI system life cycle processes," which describes additional tasks and particularities that AI brings to existing standards like ISO/IEC/IEEE 12207 and ISO/IEC/IEEE 15288. ISO/IEC 5338 contains sections that address implementation, integration, verification, and validation, to name a few, but these sections apply only to general AI development. It is up to each program to provide the appropriate foundational knowledge on AI and then consult this guidebook for details about the particularities of building AI for LAWS [ISO/IEC/IEEE 12207:2017, ISO/IEC/IEEE 15288:2023, ISO/IEC 5338:2023].

Another important document for TEVV personnel to address is DoDM 5000.101, which provides high-level guidance on model testing in Section 3.2.b.2 and on model integration testing in Section 3.2.b.3, and we recommend TEVV personnel review this guidance [DODM 5000.101]. In Chapter 6 of this guidebook, we provide an in-depth discussion on model testing. It is important to note that, while DoDM 5000.101 targets operational testing and evaluation (OT&E) and live fire testing and evaluation (LFT&E), Section 3.2 of that document specifically addresses how to support these activities across the entire acquisition lifecycle. However, it has little additional information on the testing of larger assemblies of models and associated decision-making code, and we will therefore address some of the issues in this chapter.

From the perspective of system DT&E, the tasks of model development, testing, and evaluation are like unit testing—a process that refers to testing units of the smallest code. Traditional approaches to software engineering apply unit testing through various steps of integration and eventually increase the scope of testing to the full system. This pattern holds for ML-enabled systems (MLES), and we consider ML components as another unit to be tested. See Chapter 5 for discussions about how models fit together as part of larger systems.

8.1 Recommendations for System DT&E

The following sections provide recommendations that TEVV personnel can follow to adjust their DT&E processes to the special considerations that arise when developing ML systems to function in LAWS.

8.1.1 Expand Reproducibility to System Tests

In Chapter 6, we discuss how TEVV personnel should apply reproducibility and experimental controls to models, pipelines, and tests to establish baseline testing, support diagnostics, perform debugging, and produce after-action analysis. TEVV personnel should extend this approach to system integration by exposing reproducibility and determinism configurations as well as implementing reproducibility controls in the integration code. We discuss this topic further in Section 8.2.1.

8.1.2 Capture All Test Configurations and Outputs

From an assurance perspective, it is crucially important for TEVV personnel to document necessary information about the system during testing. In Section 8.2.7, we provide details about how TEVV personnel should collect test outputs and configurations at all levels. The collected documentation not only supports the overall assurance case, but significantly benefits diagnostics, debugging, and after-mission analysis.

Development and testing teams must be prepared to collect and document a significant amount of data, and they will need corresponding amounts of accessible storage. These teams should allocate additional computational resources to analyze and study the artifacts.

8.1.3 Define an Explicit Maturation Testing Path

Over the course of development and integration, the system should evolve through the technology readiness levels [Manning 2023]. Early testing may have to rely on small amounts of synthetic data, while later stages should have access to more representative operational environments. Data and test scenarios should evolve over time, and TEVV personnel should define what those stages are early in the DT&E cycle. See Section 8.2.2 for details about this process.

8.2 Commentary on System DT&E

The following sections contain commentary about DT&E and considerations for TEVV personnel to keep in mind as they reflect on adopting the recommendations we make.

8.2.1 Reproducibility, Determinism, and Statistics

We discussed general system-level and model-level reproducibility challenges in Chapter 6, so readers can refer to that chapter for more information about how reproducibility applies to ML systems and LAWS in particular. However, for the purposes of this chapter, it is important to note that, during system development, we assume that the models themselves have already been trained and therefore we do not need to account for training variability. During runtime, due to the probabilistic nature of ML models, it is sometimes desirable for decision-making systems to choose different outcomes when these outcomes are closely matched. This technique is referred to as "temperature," and it refers to how much "risk" the algorithm takes [Wei 2024].

The main goal of standard testing practice, especially during development, is to establish a behavioral baseline. Establishing a baseline enables developers and TEVV personnel to assess, through subsequent testing, the impact that changes to code, features, environment, platform, or usage have on the system. For example, TEVV personnel can assess the impact of a new feature on general system performance such as memory. They might also use such testing to determine whether a platform change introduced a defect into one of the systems, or whether the system meets its mission goals when deployed in a new environment.

This approach doesn't fundamentally change for systems with ML, but TEVV personnel might need to adjust measurements. Ideally, the system will provide diagnostic modes and ways to control the variability of the decision logic. Basic model inference is highly repeatable and deterministic given precise thread control, and the system only needs to control the randomness injected during decision-making.

For example, PyTorch, an ML research and development platform, provides significant controls and guidance for controlling reproducibility [PyTorch 2025]. If these sorts of controls are not available, then TEVV personnel should use statistical measures to define acceptable and detailed distributions of output metrics during baseline tests. For example, documentation should include details like the acceptable minimum and maximum ranges of accuracy and the precision and recall of each class. Obviously, as with any sort of statistical measure, tighter bounds provide better indicators of variance, which can be especially useful when understanding the impacts of changes.

Ultimately, system-level testing must provide a clear understanding of how and when changes to any testing outcome are introduced, how they can be controlled for baselines, and how they impact the mission.

8.2.2 Maturing the Testing

Systems DT&E occurs along two primary dimensions. The first of these dimensions is the module path, which scales from the smallest piece of testing, such as a unit test, through various levels of integration tests, to the final system test. The second dimension involves feature maturity, and the testing along this dimension can happen independently of the end-to-end system development. If they plan for both dimensions together, TEVV personnel can perform testing more accurately in conjunction with feature development, which reduces the delays between testing and development and improves overall development productivity. The following paragraphs give some examples of how product development can progress to provide guidance for how to develop evolutionary testing.

Agile uses the term *walking skeleton* to define the earliest end-to-end version of the system that demonstrates a useful sequence of the basic features. This process is also commonly referred to as an *alpha test* where the main features pass happy path tests. Development progresses iteratively adding more features through *beta testing* and into final *gamma testing* [Cunningham 2008].

NASA has used the idea of technology readiness levels (TRL) since the 1970s to assess technology readiness, or maturity of technological components [Manning 2023]. This approach has inspired many similar readiness scales such as human readiness levels (HRL), software readiness levels (SRL), and now ML technology readiness levels (MLTRL) [Blanchette 2010; See 2021; Lavin 2022].

Like other readiness level scales, MLTRLs provide a framework for defining the various levels of maturity for ML. Each level in the document describes ML-specific needs such as data and review quality, and the levels progress gradually with each one demanding more from ML. For example, the MLTRL defines the data at level 1 as "sample data that is representative of downstream real datasets." These levels increase up to level 9, where it specifies that "Proper mechanisms for logging and inspecting data (alongside models) is critical for deploying reliable AI and ML—systems that learn on data have unique monitoring requirements" [Lavin 2022]. It is important to note that, while level 9 is about ML deployment, it is limited to single model deployment and therefore offers very little about multiple models combined with decision-making logic.

The article on MLTRLs goes on to propose *TRL cards* for ML systems, and we recommend something similar for LAWS that combines all aspects of traditional TRLs, HRLs, SRLs, and MLTRLs. Each level should also address the specific mission metrics needed for the models, the calibration levels expected, the perception demands (see Chapter 3), and the amount of repeatability and determinism expected (See section 8.2.1).

8.2.3 Reduce, Adapt, and Reuse

In Chapter 6, we discussed the fundamentals of basic model development and testing. Many of the resources we reference in that chapter address the system level and cover deployment to an operational environment. In the case of LAWS, we expect developers to develop multiple models and "deploy" them into the system development process for integration. As with the traditional MLOps process, we expect that the models (with associated data, code, and training pipelines) will need to execute to produce updated models as part of the larger system development and TEVV processes.

Reviewing existing tests from older systems is an opportunity for reuse that can prove helpful. These older tests can provide a wealth of information about how to test similar systems in an operational environment. For example, crewed weapons systems have a wide variety of safety and performance tests that should be applicable to LAWS. We anticipate that developers will create many of the early LAWS as automation systems, such as targeting support, that they will then add to an existing system. Tests from many similar systems are directly applicable for this approach, including weapons, optics, sensors, and remotely piloted vehicles. The article, "Test & Evaluation Best Practices for Machine Learning-Enabled Systems," is a good example that developers and TEVV personnel can reference to get a better picture of these processes. The article demonstrates how to use common software development processes such as combinatorial testing, differential testing, fuzz testing, and metamorphic testing on software models. Developers can also apply these strategies at the system level

[Chandrasekaran 2023]. However, developers can't apply these techniques directly to LAWS without some modification to address the special needs and difficulties of LAWS.

Finally, we want to mention *replay* because it is a common testing technique that can also prove helpful. Replay refers to a process in which TEVV personnel replay a set of recorded events through the system with injected variations to elicit defects. This type of testing is highly valuable because the recorded tests often demonstrate higher fidelity to real environments than synthetic data. The variations can serve different testing purposes such as edge case, fuzz, combinatorial, and other robustness testing without having to recapture data. In addition, TEVV personnel can easily automate and repeat replay testing to reduce the need for human interaction. MOBSTA is an example of a replay framework that can be used for ROS systems [squaresLab 2025].

8.2.4 Evolve the Tests

The evolution of testing from unit-level testing to system-integration testing may result in similar or completely different metrics and thresholds. For example, TEVV personnel might use certain accuracy metrics for a single image instance for the class tank during basic object detection. However, a different accuracy threshold may be required for a system that performs positive identification across multiple frames.

In addition, different metrics may be required for different parts of the system. For example, one part of the system might need to employ recall metrics while another might employ accuracy metrics. As TEVV personnel move to more complex parts of the PTR chain, they might also need to employ new metrics such "time in view," which requires that the system "observe" a particular target for a particular amount of time. Therefore, as TEVV personnel test higher levels of integration, they need to define, or evolve, test plans for those levels.

8.2.5 Perform Field Testing as Early as Possible

Field tests provide a level of fidelity and feedback unavailable from simulation. Therefore, they are extremely valuable for TEVV personnel to perform during development, and they should re-execute them whenever developers make changes to subsystems that interact with autonomous or ML-enabled subcomponents. MLES are complex and many problems will not emerge until integration testing occurs.

8.2.6 RAI and DIU Guidelines

The Defense Innovation Unit (DIU) provides a set of responsible AI (RAI) guidelines with worksheets for planning, development, and deployment. The questions and associated commentary in these worksheets contribute to a thoughtful and complete testing and evaluation (T&E) process across most of the lifecycle. Providing complete and detailed answers to worksheet questions can significantly improve the development process and increase the number of artifacts and outputs that TEVV personnel can use for testing as well as for the final assurance case. For example, question 4 in the DIU worksheet asks, "Have you developed an appropriate plan/interface to verify individual outputs of the system?" [Dunnmon 2021]. The verification plans that result from this planning can lead to impactful tests through all stages of maturity. TEVV personnel should pay particular attention to the commentary provided in the worksheets to make sure the planning is as thorough and useful as possible.

8.2.7 Capture and Share Data

Capturing, storing, and sharing of all telemetry, observations, and decision-making is common recommendation for complex systems for evaluation, diagnostic, and debugging purposes and is more important for MLES. MLES with decision-making components are complex and exhibit many variations in outputs due to reproducibility difficulties (see Section 8.2.1 for more information on this topic). They also experience environmental challenges, as we discussed in Chapter 3.

When we conducted human-systems integration (HSI) testing, we noted that operator trust is heavily influenced by the quantity and breadth of the testing. Capturing test data throughout the entirety of the project significantly improves the overall body of evidence that TEVV personnel can use to build assurance cases, and TEVV personnel can also use that data in longitudinal analysis to project future behavior. Capturing data properly for these purposes requires that TEVV personnel capture appropriate data with every test, such as system SBOMS, and build configurations.

In addition to following standard configuration management practices for the artifacts under testing, TEVV personnel should also use these practices when handling the collected data. TEVV personnel should begin planning a complete strategy for capturing all the development configurations and test output as early as possible, and they should reevaluate it periodically and expand it across the course of the project. TEVV personnel should make sure to include components such as the following as part of the comprehensive plan:

- The are a variety of types of *cards*—data, model, negotiation, eval, and so on—that TEVV personnel should plan to compile information for throughout the course of the project. Currently, most involve prose which creates a challenge for automated analysis, but we hope those portions of the cards will improve over time.
- The article, "Traceability for Trustworthy AI," proposes a unified, RDF-based format for containing an entire ML build process including data, SBOMs, pipelines, and the like [Mora-Cantallops 2021]. TEVV personnel should be inspired by this format.
- Mole from NIWC Pacific is an example of a system that can support the orchestration, monitoring, and data collection for autonomous experiments and test activities [Mole 2025]. Applying such a tool is a useful measure for TEVV personnel to manage data.
- The Joint Mission Environment Test Capability (JMETC) enables data collection and sharing among testing sites, and TEVV personnel should consider making use of its capabilities [JMETC 2025].
- The Test Resource Management Center (TRMC) is actively working to develop a set of AI, ML, and autonomy tools and integrate them into their extensive tools set, which would enable end-to-end integration within their test environment [TRMC 2025]. TEVV personnel should assess their usefulness and applicability to their project.

8.2.8 Security

ML introduces the need for added dimensions of security compared to traditional systems. Usually, standard practices include encryption, monitoring, IDS, and more, and these security measures should also be included in any ML system as well as the usual aspects that engineers employ to prevent cyber attacks based on system vulnerabilities, malicious code, and direct systems access. However, in addition to all these measures, developers of LAWS should consider the addition of ML support.

The deployment of ML models and the significant amount of supporting code they use can prompt many new opportunities to introduce traditional software vulnerabilities into the system. Faster processors, supporting packages, and inference engines are advancing at a rapid pace, and these require significant testing and evaluation. General cybersecurity skills and practices are well understood currently, so no new techniques are anticipated to mitigate these challenges. However, the key concern is that the amount of new code to build ML systems and the rapid change of these systems will require significant extra testing coverage.

In the following lists, we offer examples of existing security practices that developers should apply for MLES.

The following items concern measures for data security:

- **Encryption**: TEVV personnel should test and validate that all data in an ML system is encrypted with post-quantum cryptography (PQC) encryption protocols. This measure protects against unauthorized access and data leaks. Developers should document and explain any exception to this approach, such as use of non-PQC encryption protocols or completely dropping encryption.
- **Access controls**: Developers should establish a strict access control such as role-based access controls (RBAC), and they should employ authentication mechanisms to limit access controls to sensitive data and system functionality.

The following items comprise practices for security testing:

- **Penetration testing**: Development teams should schedule penetration testing at regular intervals, and it should be conducted by third-party, external experts. Penetration testing helps identify real-world vulnerabilities in the system before it is deployed.
- **Adversarial testing**: We expect that LAWS will be the target of sophisticated cyber threats, so it is crucial to evaluate how the system handles malicious inputs and tampering attempts.

The following items ensure a secure software development lifecycle:

- **Security by design**: Developers should design security into the LAWS from the beginning. Security should be a key component at every stage of the product lifecycle, including design, development, testing, and deployment.
- **Open-source software**: Developers should give priority to well-known and approved open source software.
- **Pre-tuned models**: Developers should give priority to preapproved, pretested, and hardened models.

8.2.9 Runtime Measures are Still Applicable

Approaches to and techniques for addressing normal concerns, such as runtime performance and memory utilization, remain consistent when applied to the development of AI, but they can vary greatly in detail. For example, while ML models often require significantly more resources than systems without ML components, the same techniques and measures apply when identifying resource and timing usage.

9 System OT&E

This chapter provides guidance for personnel that perform operational testing and evaluation (OT&E) and live-fire testing and evaluation (LFT&E) of machine learning- (ML)-enabled lethal autonomous weapons systems (LAWS).

There are recent advances in the policy from DOT&E in DoD Manual 5000.100 and DoD Manual 5000.101 [DODM 5000.100, DODM 5000.101]. These documents, combined with guidance from the Defense Innovation Unit (DIU) in their Deployment Worksheet, provide a good starting point for the general deployment and the testing and evaluation (T&E) of LAWS [Dunnmon 2021].

All testing is highly dependent on how well developers specify and design the system. Consequently, this chapter reflects and resonates with concepts in Chapter 3 and Chapter 5, and we highly recommend reading those chapters before this one. In this chapter, we go beyond policy and higher-level guidance to provide specific detailed recommendations and observations about how to obtain test data, develop operational tests, create multi-model interactions, develop ML operations (MLOps) and process interactions, field updates, prepare continuous learning, and mitigate counter artificial intelligence (AI).

The introduction of new technologies such as ML-enabled systems (MLES) and LAWS challenge the approach to “test as we fight.” These systems introduce new capabilities and procedural changes to how we and our adversaries operate, which means that the notion of “as we fight” is constantly evolving. This evolution increases the need for risk identification, especially with regard to emergent behaviors and human-systems interaction (HSI) in the testing of LAWS. Additionally, novel fighting methods and corresponding reactions emphasize the responsible AI (RAI) perspective on governability and the need for understandability in the systems.

As mentioned throughout this guidebook, data, and consequently sensing, both play a central role in differentiating how developers build MLES in contrast to existing weapons systems. Understanding how ML systems sense and analyze the environment becomes the focus of the differences between the T&E of LAWS and traditional weapons systems. Chapter 3 offers in-depth discussions about the operational context for these systems and their defining requirements. A key challenge for T&E is constructing a relevant operational test environment with sufficient fidelity to support the new operational uses identified by these requirements.

During our initial evaluation of user trust, we repeatedly heard that these systems required extensive testing, especially for user interaction [Hale 2025]. Users expect these systems to be thoroughly tested, and they want evidence of the breadth, depth, and quantity of this testing. From an OT&E perspective, we expect an increase in demand across all dimensions of testing. DODM 5000.101 emphasizes the use of science and technology approaches for testing, heavy use of automation, and extensive testing with humans [DODM 5000.101]. As described above, the novelty of the new capabilities and testing approaches will require the allocation of additional time and resources.

9.1 Recommendations for System OT&E

The following sections outline our recommendations for the OT&E of LAWS.

9.1.1 Allocate Significant Additional Time and Effort to Testing

In some disciplines, testing and quality control personnel can assess quality by using well-established and well-understood techniques to evaluate the materials, designs, and processes involved in making products, and the time it takes to test the final device comprises only a small fraction of the final effort. In comparison, MLES are not well understood, and the engineering discipline that underpins ML development activities is still developing, so it follows that testing activities are more involved and time-consuming than they would be for traditional systems that are better understood and less complex. As mentioned above, thorough testing of MLES is also necessary to engender user trust in these systems. While it is so common for TEVV personnel to ask for more time and resources for testing that the request is often disregarded, the need for more time and resources in this case is legitimate, and development teams should take it seriously and plan for it accordingly.

9.1.2 Capture Datasets for Evaluation and Replay

Operational test environments can provide entirely new sources of test "data" in the form of captures and traces from experiments. Conditions that arise during new tests, even those for use only during development, will always introduce randomness in the environment through changes to weather, locations, users, and other stochastic elements that can elicit new risks. This phenomenon is true for ML-enabled LAWS because the ML components perform with probabilistic outputs measured against thresholds. Therefore, slight changes in inputs can have significant changes to outputs at boundary conditions. To address these issues, TEVV personnel should capture sensor data, events, decisions, and other traces during these tests to perform significant post-test analysis and replay testing. We discuss these issues in further detail in Section 9.3.1.

9.2 Observations on System OT&E

The following sections offer observations about the recommendations we offer above.

9.2.1 Testing a Model Is Not the Same as Testing a System

Many tools, papers, and approaches focus on single-model testing including both standalone or incorporated. The accuracy of an ML model, which shows the general accuracy of predictions, does not reflect the accuracy of an AI system, which includes the surrounding, safety-governed, decision-making logic. We discuss the difference between testing at the model level as opposed to testing at the system level in Section 9.3.1, Chapter 3, and Chapter 5.

9.2.2 Data Is a Small Reflection of the Test Environment

Advanced autonomous systems use multiple sensors and models to understand their environment and make decisions, and the number of sensors and models that systems tend to use is rapidly increasing. Test datasets may only provide small perspectives on the world and do not represent all the conditions that the system might encounter. We discuss these issues further in Section 9.3.1 on test datasets and in Section 9.3.5 on model reliability and robustness.

9.2.3 Decision-Making Systems Require Greater Resources to Develop and Test than Previous Systems

By their nature and because of their purpose, autonomous systems perform more frequent and more complex decision-making than previous systems. Recording and analyzing data about this decision-making requires additional data storage, analysis time, and analysis skills, all of which require more testing time to ensure these elements work properly. We offer further discussion about the additional resources that development teams need to build these elements in Section 9.3.7, and we also discuss the importance of transparency and observability for these elements.

9.3 Commentary on System OT&E

The following sections provide commentary about how to carry out the recommendations we outline above.

9.3.1 Test Datasets and Model Interaction

In Chapter 3 and Chapter 5, we describe how fully autonomous LAWS involve the integration of many different sensors and algorithms in complex and dynamic environments. Because of the complexity of those environments, there can be no single, definable dataset that—on the one hand—represents all the conditions that ML components will face and that—on the other hand—TEVV personnel can use to understand the interactions of the ML components within the overall system.

When developing operational test plans, it is important for TEVV personnel to be aware of how the environmental conditions impact each piece of ML and the symbolic logic, and how the test conditions need to be varied to cover the necessary permutations. DoDM 5000.101 section 3.2.b.2.c recommends mapping all capabilities to operational and system requirements [DODM 5000.101]. As with any testing, full combinatorial testing is intractable, and understanding the ML components and their relationships allows for smarter pairwise and n-way test configuration, but how to broadly map capabilities to requirements is unclear. For example, many object detectors work on static frames, but tracking algorithms work on the relationship between frames. Frame-to-frame variables such as framerate, therefore, will impact tracking differently but will not impact single frames for an object detector. Developers and TEVV personnel should use many of the same considerations and thinking that went into writing (or backfilling) requirements when understanding the basic qualities. We discuss the various complexities in the perception, targeting, and response (PTR) chain in Section 3.3.7.

After TEVV personnel develop and perform preliminary operational tests, they should analyze the data they captured for alignment with test data, and they can use replay mechanisms to identify potential failures. TEVV personnel can then use the findings from these activities to develop new operational tests. For example, TEVV personnel can capture a *rosbag* on a Robot Operating System (ROS), which captures all the events for playback [ROS 2025]. Tools such as MOBSTA can play back the rosbag and generate artificial permutations in the input stream to see system impacts in simulation, reducing cost [squaresLab 2025]. These tools are often employed during DT&E but have their place in OT&E for rapidly developing better operational tests based on the operational test range of the captured data. As with traditional real-time systems, timing between components can have tremendous impacts on autonomy and system performance that only data captured through operational testing can fully reveal. TEVV personnel should retain full test captures for future reference to support activities such as diagnostics and field updates.

Section 3.2 of DoD Manual 5000.101 goes into detail about data management, model testing, and integrated model tests. As mentioned above, *creating* a single dataset for the operational testing is intractable, but using captured representative traces is an option to avoid this challenge. TEVV personnel should apply the guidance in DoDM 5000.101 to any captured operational datasets using the same criteria they would apply anywhere, including training [DODM 5000.101].

9.3.2 Continuous Learning and Emergent Behavior

We do not discuss continuous learning systems as part of this guidebook because they are beyond its scope. However, the DoD has shown the intention to use continuous learning systems to augment AI capabilities for LAWS. For example, Section 3.2.b.3.f of DODM 5000.101 describes systems capable of "Characterize[ing] […] emergent behavior and negative test results as the AI-enabled or autonomous DoD system learns and changes its behavior based on the input data it operates within" [DODM 5000.101]. That passage describes capabilities that are like the kind provided by continuous learning systems or by a kind of runtime adjustment that can change system behavior over time. Because the passage implies the use of continuous learning systems, we thought it necessary to provide some commentary about them in this section even though we don't cover them in detail in this guidebook.

First, we want to stress that, if continuous learning is part of the operation system, TEVV personnel should prepare for significant, even major, impact on testing effort. The reason for the increase in testing effort is that cumulative side effects of systems, whether intentional or accidental, add significant complexity to testing. To determine long-term impacts from use, TEVV personnel must develop test plans that evaluate factors such as when to restart and reinitialize, how long to run tests, what kind of impacts might occur between tests, how to sequence tests, and so on. After diagnosing failures and risk, TEVV personnel must then consider all the previous activity when developing reproducibility steps and understanding potential operational impact.

Designing tests for continuous learning or self-adaptive systems will also require more effort. TEVV personnel must consider how the effects of one test will affect future tests. Tests are no longer independent, and changes to the model can be obvious or subtle. In addition, TEVV personnel might need to consider performing complex testing for counter-AI strategies such as data poisoning. However, a full discussion of how to construct well-sequenced tests and the impact of continuous learning is currently beyond the scope of this guidebook.

9.3.3 Field Updates

Over the course of operation, it will be important to update the system's ML models as well as its software. ML models may need updates for several reasons such as to deploy increased capability, to fix defects, or to patch vulnerabilities. Depending on what changes developers want to deploy, approval for updates may require a full senior review of the system based on the requirements of DODD 3000.09. If the updates significantly change the scope of the system's functions, then TEVV personnel might need to devise additional tests, and they might need to fully re-execute operational testing.

When testing minor updates, the test plans may need only minor alterations based on the intended changes, such as adjustments to accuracy or recall metrics. TEVV personnel can replay captured test runs (as we discussed in Section 9.3.1) to identify variations that the updates introduced, and TEVV personnel can also use those results to identify new testing needs.

TEVV personnel should also test the features of field updates and rollback. ML models are like any software, so TEVV personnel should test the access required to modify the system as well as for security vulnerabilities and other system-access vulnerabilities. Updates to ML models may be larger than updates for other systems, and development teams should therefore allocate additional time for testing than they would allocate for traditional software systems.

9.3.4 Iteration and MLOps

DevOps is the common term for the tight, iterative integration of development and operations. This tightly integrated and iterative model has progressed through many variations and expansions over the years, including a version for ML known as MLOps.

Like field updates, MLOps aims for tight integration of model development and operational deployment. DODM 5000.101 does not identify MLOps directly, but the document mentions concepts similar to those practiced by MLOps. For example, in Section 3.2.a, it states that "the acquisition life cycle must keep pace with the development cadence of the AI model." In Section 3.2.b.2.d, the document states the requirement to "Repeat[s] […] the training phase and its V&V until the trained model reaches the required performance" [DODM 5000.101]. While these requirements are similar to those for field updates, developers must also consider the pipeline itself and development process as part of the operational model when implementing an MLOps approach. In these cases, training, retraining, tuning and testing—which were traditionally DT&E and contractor activities—may now form part of the operational model. Until such time as fully integrated testing (i.e., through the integration of CT, DT, and OT) exists, developers and TEVV personnel should consider adopting these development-like activities.

9.3.5 Reliability and Robustness: Uncertainty Quantification

Under the current state-of-the-practices for ML, trained models are not directly calibrated to the data set they are trained against and must be calibrated after training. TEVV personnel should look for evidence of model calibration and what particular type of model calibration development teams performed. *A Guide to Failure in Machine Learning: Reliability and Robustness from Foundations to Practice* provides a detailed examination of the subject [Heim 2025]. The GitHub project "Generalized Calibration Error Python Package" provides an example Python tool that provides a way to determine if a model has been calibrated. We recommend that TEVV personnel make use of such a tool [GCE 2025].

9.3.6 Counter AI

Counter AI refers to attacks on AI systems that employ techniques for making the AI "learn the wrong thing," "do the wrong thing," or "reveal the wrong thing"—all of which we discuss in the sections that follow. However, as a side note, we want to make an important distinction: the processes associated with preventing or mitigating counter AI should not be confused with security. General security practices for software intensive systems comprise a different area and, therefore, a different set of considerations, and we want to stress that developers should implement all of them for use in ML-enabled LAWS.

The whitepaper, *Counter AI: What Is It and What Can You Do About It?* offers a good place to start thinking about counter AI. In that whitepaper, the authors provide an overview of where AI attacks occur in the technology stack, and they introduce the three attack types that we mentioned above: *learn the wrong thing, do the wrong thing,* and *reveal the wrong thing*. They also discuss five different threat models [VanHoudnos 2024]. We provide an overview of that whitepaper in this section to introduce and discuss some of its terms and how counter AI can impact LAWS and the OT&E process.

Learn the wrong thing attacks occur during training or retraining, generally as a type of modification to the inputs affecting a specific part of the system or function of the system that attackers want to disrupt. These inputs usually target a narrow, mission-specific purpose. For example, the result of such attacks might result in the system misclassifying a particular military asset as civilian. These input modifications may be subtle and therefore not obvious to human inspectors. During normal training, this attack might progress through a compromised supply chain or through another path that results in the modification of the source data, perhaps even though an attack on the training pipeline. While this type of attack occurs during what is normally part of the development lifecycle, it can also occur during tuning or retraining when preparing updates to the system. TEVV personnel should therefore test the system when any change occurs to the model or system as we discussed in section 9.3.4. Emerging research shows that, for certain attack types, even small amounts of data can significantly impact the performance of the model. If the system employs continuous learning or adaptive systems, then the problem can become significantly more complex. Preventing or mitigating this type of attack for systems with continuous learning or adaptive systems is beyond the current scope of this guidebook.

Do the wrong thing attacks on LAWS occur by modifying the sensor inputs to exploit weaknesses in the ML algorithm. This attack can progress by means of a cybersecurity attack or changes to real-world artifacts perceived by the sensors where attackers make changes to the inputs that the software is assessing. For example, attackers discovered during the early development of self-driving cars that placing tape on a stop sign in a certain pattern caused the car's software to misidentify the stop sign as something else. TEVV personnel can test for these types of attacks during security testing by making various modifications to input data such as partial obstruction, smudging, or other data distortions. The JATIC toolset provides some tools to help develop counter-AI test sets [CDAO 2025].

Another method for perpetrating this type of attack is through the placement of specifically created physical items, often referred to as *patches*, in the environment, which cause the ML to function incorrectly [Sharif 2016]. Construction of effective patches requires an understanding of the ML models and the system to some degree as well as the right tools and significant time.

Finally, *reveal the wrong thing* attacks—often referred to as model inversion or membership inference attacks—involve getting the ML model to reveal data about how it is trained, or even to reveal specific images. The information that attackers extract can be particularly useful when trying to understand how and why a model was developed. This type of attack usually requires the ability to repeatedly interrogate a model, which is not an issue with a weapon system unless the model is extracted from the weapon system through a cyber attack or through direct access.

9.3.7 Transparency, Observability, and Governability

One of the key ways in which LAWS are different from other weapons systems is that they make many decisions autonomously, and, crucially, humans must understand how and why they made them. Systems, therefore, must be fully transparent about how they made decisions, providing clarity about each step of the decision-making process from acquiring initial data from sensors through the PTR chain to final fire control. Knowing why the device behaved the way it did and how directions from operators can affect that decision-making process is fundamental to developing operator trust.

DoDD 3000.09 specifies a variety of ways the systems should be transparent and auditable by relevant personnel during design, development, and testing as well as during use [DoDD 3000.09]. Previously, OT&E testing was less about observing feedback and promoting transparency during development except as a source of evidence, but this focus will change with the inclusion of iterative processes and the potential for field updates. For example, understanding the properties of additional data for retraining or tuning, and what biases might exist in it, adds additional demands on the T&E process, making transparency crucial to the process.

Ultimately, development and testing processes should expose how configuration settings translate into operator control and changes to system behavior. Chapter 5 of this document and the *Reference Architecture for Ethical Conduct in LAWS* describe how different controls introduced to a system architecture ultimately affect its behavior [Mellinger 2025]. Developers should make these pathways visible in design documents and through the various monitoring tools and logging tools.

References

URLs are valid as of the publication date of this report.

[ACM 2020]
Artifact Review and Badging. *Associate for Computing Machinery (ACM) Website*. August 2020. https://www.acm.org/publications/policies/artifact-review-and-badging-current

[Afzal 2020]
Afzal, Afsoon; Goues, Claire Le; Hilton, Michael and Timperley, Christopher Steven. A Study on Challenges of Testing Robotic Systems. Pages 96-107. *2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST).* October 2020, https://doi.org/10.1109/ICST46399.2020.00020

[Bass 2021]
Bass, Len; Clements, Paul & Kazman, Rick, *Software Architecture in Practice, 4th Edition.* Addison-Wesley Professional. 2021. 978-0-13-688609-9. https://www.pearson.com/en-us/subject-catalog/p/software-architecture-in-practice/P200000000111/9780137468218

[Bass 2025]
Bass, Len; Lu, Qinghua; Weber, Ingo & Zhu, Liming. *Engineering AI Systems: Architecture and DevOps Essentials*. Addison-Wesley Profession. March 2025. ISBN-13 978-0138261412. https://www.pearson.com/en-us/subject-catalog/p/engineering-ai-systems-devops-and-architecture-approaches/P200000011757/9780138261450

[Blanchette 2010]
Blanchette, Jr., Stephen; Albert, Cecilia; & Miller, Suzanne. *Beyond Technology Readiness Levels for Software: U.S. Army Workshop Report.* CMU/SEI-2010-TR-044. Software Engineering Institute. 2010. https://insights.sei.cmu.edu/library/beyond-technology-readiness-levels-for-software-us-army-workshop-report/

[Bower-Sinning 2024]
Brower-Sinning, Rachel; Lewis, Grace A.; Echeverría, Sebastián & Ozkaya, Ipek. Using Quality Attribute Scenarios for ML Model Test Case Generation. Pages 307-310. *2024 IEEE 21st International Conference on Software Architecture Companion (ICSA-C)*. June 2024. https://arxiv.org/abs/2406.08575

[Brooke 2013]
Brooke, John. SUS: A retrospective. *Journal of User Experience*. Volume 8, Issue 2. February 2013. Page 29-40. https://uxpajournal.org/sus-a-retrospective

[CDAO 2024]
CDAO. *Test and Evaluation of Artificial Intelligence Models.* CDAO. April 2024. https://www.ai.mil/Portals/137/Documents/Resources%20Page/Test%20and%20Evaluation%20of%20Artificial%20Intelligence%20Models%20Framework.pdf

[CDAO 2024b]
CDAO. *Human Systems Integration Test and Evaluation of Artificial Intelligence Enabled Capabilities: What to Consider in a Test & Evaluation Strategy.* CDAO. April 2024. https://www.ai.mil/Portals/137/Documents/Resources%20Page/Human%20Systems%20Integration%20Test%20and%20Evaluation%20of%20AI-Enabled%20Capabilities%20Framework.pdf

[CDAO 2025]
Welcome to the JATIC docs! *CDAO JATIC Documentation Website*. March 4, 2025. [accessed]. https://cdao.pages.jatic.net/public/

[Chandrasekaran 2023]
Chandrasekaran, Jaganmohan; Cody, Tyler; McCarthy, Nicola; Lanus, Erin & Freeman, Laura. Test & Evaluation Best Practices for Machine Learning-Enabled Systems. 2023. arXiv. https://arxiv.org/abs/2310.06800

[Clemens-Sewall 2024]
Clemens-Sewall Mary Versa; Rafkin, Emma and Cervantes, Christopher. Domain Knowledge Eli for Data Curation to Promote Trustworthiness in Artificial Intelligence. Pages 21-30. In *Proceedings of the 2024 International Conference on Assured Autonomy (ICAA).* 2024 Oct 10. https://ieeexplore.ieee.org/abstract/document/10765964

[Clemens-Sewall 2025]
Clemens-Sewall, Mary Versa; Cervantes, Christopher; Rafkin, Emma; Otte, J. Neil; Magelinski, Tom; Lewis, Libby; Liu, Michelle; Udwin, Dana & Kirkman-Bey, Monique. CaTE Data Curation for Trustworthy AI. Carnegie Mellon University and JHU APL, March 2025. [Link Pending]

[Clements 2011]
Clements, Paul. Improving Testing Outcomes Through Software Architecture [blog post]. *SEI Insights*. August 2011. https://insights.sei.cmu.edu/blog/improving-testing-outcomes-through-software-architecture/

[Comella-Dorda 2004]
Comella-Dorda, Santiago; Dean, John; Lewis, Grace; Morris, Edwin; Oberndorf, Patricia; & Harper, Erin. *A Process for COTS Software Product Evaluation.* CMU/SEI-2003-TR-017. Software Engineering Institute. 2004. https://doi.org/10.1184/R1/6571721.v1

[Cox 2016]
Cox, James C.; Kerschbamer, Rudolf; & Neururer, Daniel. "What is trustworthiness and what drives it?" *Games and Economic Behavior*. Volume 98. July 2016. Pages 197-218. https://doi.org/10.1016/j.geb.2016.05.008

[Cunningham 2008]
Cunningham, Ward. *Walking Skeleton*. August 22, 2008. https://wiki.c2.com/?WalkingSkeleton

[CVAT 2025]
Computer Vision Annotation Tool (CVAT). *GitHub*. March 27, 2024. [accessed]. https://github.com/cvat-ai/cvat

[DAU 2024]
DAU. DoD AI Definition Reference Chart. DAU. December 2024. https://www.dau.edu/sites/default/files/Migrated/CopDocuments/DAI%20AI%20Toolkit%20-%20AI%20Descriptors%20and%20Definitions%20Reference%20Charts.pdf

[DoD 2021a]
Deputy Secretary of Defense. Memorandum: Implementing Responsible Artificial Intelligence in the Department of Defense. Department of Defense. May 2021. https://media.defense.gov/2021/May/27/2002730593/-1/-1/0/IMPLEMENTING-RESPONSIBLE-ARTIFICIAL-INTELLIGENCE-IN-THE-DEPARTMENT-OF-DEFENSE.PDF

[DoD 2022a]
DoD Responsible AI Working Council. *Responsible Artificial Intelligence Strategy and Implementation Pathway.* Department of Defense. June 2022. https://media.defense.gov/2022/Jun/22/2003022604/-1/-1/0/Department-of-Defense-Responsible-Artificial-Intelligence-Strategy-and-Implementation-Pathway.PDF

[DoDD 3000.09]
Office of the Under Secretary of Defense for Policy. *Autonomy in Weapon Systems*. DoD Directive 3000.09. January 2023 https://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodd/300009p.pdf?ver=2019-02-25-104306-377

[DoDI 3000.17]
Office of the Under Secretary of Defense for Policy. *Civilian Harm Mitigation and Response.* DoD Instruction 3000.17. December 21, 2023. https://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodi/300017p.pdf

[DoDI 5000.98]
Office of the Director, Operational Test and Evaluation. *Operational Test and Evaluation and Live Fire Test and Evaluation.* DOD Instruction 5000.98. December 9, 2024. https://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodi/500098p.PDF

[DoDM 5000.100]
Office of the Director, Operational Test and Evaluation. *Test and Evaluation Master Plans and Test and Evaluation Strategies.* DOD Manual 5000.100. December 9, 2024. https://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodm/5000100m.PDF

[DoDM 5000.101]
Office of the Director of Operational Test and Evaluation. *Operational Test and Evaluation and Live Fire Test and Evaluation of Artificial Intelligence-Enabled and Autonomous Systems.* DOD Manual 5000.101. December 9, 2024. https://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodm/5000101p.PDF

[Dunnmon 2021]
Dunnmon, Jared; Goodman, Bryce; Kirechu, Peter; Smith, Carol & Van Deusen, Alex. *Responsible AI Guidelines in Practice*. Defense Innovation Unit. 2021. https://www.diu.mil/responsible-ai-guidelines

[Gardner 2023]
Gardner, Carrie; Robinson, Katherine-Marie; Smith, Carol & Steiner, Alexandrea. Contextualizing End-User Needs: How to Measure the Trustworthiness of an AI System [blog post]. *SEI Blog* (blog). July 17, 2023. https://doi.org/10.58012/8b0v-mq84.

[GCE 2025]
Generalized Calibration Error Python Package. *GitHub.* April 2025. [accessed]. https://github.com/cmu-sei/gce

[Gray 2016]
Gray, Doug. *Applying the Goal-Question-Indicator-Metric (GQIM) Method to Perform Military Situational Analysis*. Software Engineering Institute. 2016. https://insights.sei.cmu.edu/library/applying-the-goal-question-indicator-metric-gqim-method-to-perform-military-situational-analysis/

[Hale 2025]
Hale, Matt. *Measuring Trust: Concept Testing and User Trust Evaluation in Autonomous Systems*. Software Engineering Institute. 2025. [Link Pending]

[Harper 2023]
Harper, John. Pentagon to launch pilot focused on "calibrated trust" in AI. *DefenseScoop*. August 29. 2023. https://defensescoop.com/2023/08/29/pentagon-to-launch-pilot-focused-on-calibrated-trust-in-ai/

[Hart 1986]
Hard, Sandra G. *NASA Task Load Index (TLX): Paper and Pencil Package – Volume 1.0.* NASA 1986. https://ntrs.nasa.gov/citations/20000021488

[Heim 2025]
Heim, Eric; Wright, Oren; & Shriver, David. *A Guide to Failure in Machine Learning: Reliability and Robustness from Foundations to Practice.* March 2025. https://arxiv.org/abs/2503.00563

[Hicks 2023]
Hicks, Kathleen. *Data, Analytics, and Artificial Intelligence Adoption Strategy*. Department of Defense. 2023. https://media.defense.gov/2023/Nov/02/2003333300/-1/-1/1/DOD_DATA_ANALYTICS_AI_ADOPTION_STRATEGY.PDF.

[Horneman 2019]
Horneman, Angela; Mellinger, Andrew & Ozkaya, Ipek. *AI Engineering: 11 Foundational Practices*. Software Engineering Institute. 2019. https://insights.sei.cmu.edu/documents/582/2019_019_001_634648.pdf

[Hugging Face 2025]
Model Cards. *Hugging Face.* January 30, 2025. https://huggingface.co/docs/hub/en/model-cards

[Huyen 2022]
Huyen, Chip. Designing Machine Learning Systems: An Iterative Process for Production-ready Applications. ISBN: 9781098107963. O'Reilly Media, Incorporated. 2022. https://www.oreilly.com/library/view/designing-machine-learning/9781098107956

[IEEE 7000-2021]
IEEE. *IEEE Standard Model Process for Addressing Ethical Concerns during System Design.* IEEE Std 7000-2021. September 2021. https://doi.org/10.1109/IEEESTD.2021.9536679

[IEEE 7001-2021]
IEEE. *IEEE Standard for Transparency of Autonomous Systems.* IEEE Std 7001-2021. December 2021. https://doi.org/10.1109/IEEESTD.2022.9726144

[IEEE 2025]
IEEE Code of Ethics. *IEEE Website*. April 7, 2025. [accessed]. https://www.ieee.org/about/corporate/governance/p7-8.html

[ImageNet 2025]
ImageNet. *ImageNet Website*. March 3, 2025. [accessed]. https://www.image-net.org/

[Institute for Ethical AI & Machine Learning 2025]
The Responsible Machine Learning Principles. *The Institute for Ethical AI & Machine Learning Website*. February 25, 2025. [accessed]. https://ethical.institute/principles.html

[ISO/IEC 5338:2023]
ISO/IEC. *Information technology — Artificial intelligence — AI system life cycle processes.* ISO/IEC Standard 5338:2023. December 2023. https://www.iso.org/standard/81118.html

[ISO/IEC/IEEE 12207:2017]
ISO/IEC/IEEE. Systems and software engineering – Software life cycle processes. ISO/IEC Standard 12207:2017. November 2017. https://www.iso.org/standard/63712.html

[ISO/IEC/IEEE 15288:2023]
ISO/IEC/IEEE. Systems and software engineering – System life cycle processes. ISO/IEC Standard 15288:2017. May 2023. https://www.iso.org/standard/81702.html

[Jeffries 2022]

Jeffries, Daniel. Why We Started the AIIA and What It Means for the Rapid Evolution of the Canonical Stack of Machine Learning. *AI Infrastructure Alliance.* Jan 7, 2022. https://ai-infrastructure.org/why-we-started-the-aiia-and-what-it-means-for-the-rapid-evolution-of-the-canonical-stack-of-machine-learning/

[JMETC 2025]

Joint Mission Environment Test Capability (JMETC). *JMET Website*. March 11, 2025. [accessed]. https://www.trmc.osd.mil/jmetc-home.html

[Joseph 2022]

Joseph, VR. Optimal ratio for data splitting. *Statistical Analysis and Data Mining: The ASA Data Science Journal*. Volume 15. Number 4. August 2022. Page 531-538. https://onlinelibrary.wiley.com/doi/full/10.1002/sam.11583

[JP 3-60]

Chairman of the Joint Chiefs of Staff. *Joint Targeting.* Joint Publication 3-60. September 2018. https://www.esd.whs.mil/Portals/54/Documents/FOID/Reading%20Room/Joint_Staff/21-F-0520_JP_3-60_9-28-2018.pdf

[Kästner 2025]

Käster, Christian. *Machine Learning in Production: From Models to Products.* MIT Press. April 2025. https://mlip-cmu.github.io/book/

[Krafft 2020]

Krafft, Tobias; Hauer, Marc; Fetic, Lajla; Kaminski, Andreas; Puntschuh, Michael; Otto, Philipp; et al. *From Principles to Practice - An interdisciplinary framework to operationalise AI ethics*. Artificial Intelligence Ethics Impact Group. April 2020. https://www.bertelsmann-stiftung.de/fileadmin/files/BSt/Publikationen/GrauePublikationen/WKIO_2020_final.pdf

[Lacey 2000]

Lacey, Mike O.; Bill, Brian J.; Berrigan, Michael J.; Boehman, Michael P. & Chiarella, Louis A. *Operational Law Handbook*. Judge Advocate General's School Charlottesville VA. 2000. https://apps.dtic.mil/sti/citations/tr/ADA377522

[Martinez 2024]

Martinez , David R. & Kifle, Bruke M. *Artificial Intelligence: A Systems Approach from Architecture Principles to Deployment*. MIT Lincoln Laboratory Series. The MIT Press 2024. https://doi.org/10.7551/mitpress/14806.001.0001

[Maffey 2023]

Maffey, Katherine R; Dotterrer, Kyle; Niemann, Jennifer; Cruickshank, Iain; Lewis, Grace A & Kästner, Christian. MLTEing models: Negotiating, Evaluating, and Documenting Model and System Qualities. Pages 31-36. *IEEE/ACM 45th International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*. May 2023. https://arxiv.org/abs/2303.01998

[Mitchell 2018]
Mitchell, Margaret; Wu, Simone; Zaldivar, Andrew; and Barnes, Parker; Vasserman, Lucy; Hutchinson, Ben; Spitzer, Elena; Raji, Inioluwa Deborah & Gebru, Timnit. *Model Cards for Model Reporting*. Proceedings of the Conference on Fairness, Accountability, and Transparency, pages 220-229. ACM. 2018 https://arxiv.org/abs/1810.03993

[MLTE 2025]
MLTE Documentation. January 30, 2025. [accessed]. https://mlte.readthedocs.io/en/latest/

[Mole 2025]
Mole. *GitHub*. March 11, 2025. [accessed]. https://github.com/niwcpac/mole

[squaresLab 2025]
squaresLab. MOBSTA. March 10, 2025. [accessed]. https://github.com/squaresLab/MOBSTA

[Mora-Cantallops 2021]
Mora-Cantallops, Marçal; Sánchez-Alonso, Salvador; García-Barriocanal, Elena & Sicilia, Miguel-Angel. Traceability for Trustworthy AI: A Review of Models and Tools. *Big Data and Cognitive Computing.* Volume 5, number 2. 2021. https://www.mdpi.com/2504-2289/5/2/20

[Lavin 2022]
Lavin, Alexander; Gilligan-Lee, Ciarán M.; Visnjic, Alessya; Ganju, Siddha; Newman, Dava; Ganguly, Sujoy; Lange, Danny; Baydin, Atílím Güneş; Sharma, Amit; Gibson, Adam; Zheng, Stephan; Xing, Eric P.; Mattmann, Chris; Parr, James & Gal, Yarin. *Technology readiness levels for machine learning systems.* Nature Communications 13, Article 6039 (2022). https://doi.org/10.1038/s41467-022-33128-9. https://doi.org/10.1038/s41467-022-33128-9

[Mellinger 2025]
Mellinger, Andrew O, et.al *Reference Architecture for Assuring Ethical Conduct in LAWS.* Software Engineering Institute. 2025. [TODO: DOI Link Pending.]

[Mellinger 2025b]
Mellinger, Andrew; Justice, Daniel; Connor, Marissa; Gallagher, Shannon & Brooks, Tyler. The Myth of Machine Learning Reproducibility and Randomness for Acquisitions and Testing, Evaluation, Verification, and Validation [blog post]. *SEI Blog*. January 13, 2025. https://doi.org/10.58012/g17y-gp09.

[Manning 2023]
Manning, Catherine G. Technology Readiness Levels. *NASA Website.* September 27, 2023. https://www.nasa.gov/directorates/somd/space-communications-navigation-program/technology-readiness-levels/

[NSLD 2022]
National Security Law Department (NSLD). *Law of Armed Conflict Documentary Supplement.* The Judge Advocate General's Legal Center & School. 2022. https://www.loc.gov/item/2009210362

[Nichols 2023]
Nichols, Bill; Weinstock, Chuck; Goodenough, John; Woody, Carol & Ellison, Bob. *GQIM and Assurance Cases*. Software Engineering Institute. 2023. https://apps.dtic.mil/sti/citations/trecms/AD1193672

[NIST 2024]
NIST. "Assurance." *NIST Computer Security Resource Center Website*. December 2, 2024 [accessed]. https://csrc.nist.gov/glossary/term/assurance

[NIST 2025]
NIST. "Trust." *NIST Computer Security Resource Center Website*. February 10, 2025 [accessed]. https://csrc.nist.gov/glossary/term/trust

[Norquist 2020]
Norquist, David. *DoD Data Strategy*. Department of Defense. 2020. https://media.defense.gov/2020/Oct/08/2002514180/-1/-1/0/DOD-DATA-STRATEGY.PDF.

[NSSF 2024]
4 Primary Rules for Gun Safety. *The Firearm Industry Trade Association Website.* December 10, 2024 [accessed]. https://www.nssf.org/articles/4-primary-rules-of-firearm-safety/

[Nvidia 2024]
PeopleNet Model Card. *Nvidia Website*. November 12, 2024. https://catalog.ngc.nvidia.com/orgs/nvidia/teams/tao/models/peoplenet

[Mohseni 2022]
Mohseni, Sina, Haotao Wang, Chaowei Xiao, Zhiding Yu, Zhangyang Wang, and Jay Yadawa. Taxonomy of machine learning safety: A survey and primer. *ACM Computing Surveys.* Volume 55. Number 8. December 2022. Pages 1-38. https://dl.acm.org/doi/full/10.1145/3551385

[OECD 2025]
OECD. Catalogue of Tools & Metrics for Trustworthy AI (Metrics Tab). *OECD.AI Policies, data and analysis for trustworthy artificial intelligence*. 2024. https://oecd.ai/en/catalogue/metrics

[OGC 2023]
Office of General Counsel, Department of Defense. *Department of Defense Law of War Manual.* July 2023. https://ogc.osd.mil/Portals/99/Law%20of%20War%202023/DOD-LAW-OF-WAR-MANUAL-JUNE-2015-UPDATED-JULY%202023.pdf

[Olechowski 2020]
Olechowski, Alison L; Eppinger, Steven D; Joglekar, Nitin & Tomaschek, Katharina. Technology readiness levels: Shortcomings and improvement opportunities. Systems engineering: the journal of the International Council on Systems Engineering, 2020, Vol.23(4), p.395-408. https://incose.onlinelibrary.wiley.com/doi/10.1002/sys.21533

[Pushkarna 2022]
Pushkarna M, Zaldivar A, Kjartansson O. Data cards: Purposeful and transparent dataset documentation for responsible ai. Pages 1776-1826. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*. 2022 Jun 21. https://arxiv.org/abs/2204.01075

[PyTorch 2025]
Reproducibility. *PyTorch Website*. March 10, 2025. [accessed]. https://pytorch.org/docs/stable/notes/randomness.html

[Rigsbee 2025]
S. Rigsbee, S; Felsen, C. R.; Kerbel, K. R.; Kang, R. M. & Camacho, B. Human-Centric, Teaming-Focused Approach for Design and Development of Non-Deterministic Systems : A Human Machine Teaming Design Framework. Carnegie Mellon University. March 2025. [link pending]

[Roboflow 2024]
Roboflow. What is YOLO? The Ultimate Guide [2025] [blog post]. *Roboflow blog*. January 9, 2025. https://blog.roboflow.com/guide-to-yolo-models

[ROS 2025]
Rosbag. *Robot Operating System (ROS) Website*. March 11, 2025. [accessed]. http://wiki.ros.org/rosbag

[Rosebrock 2020]
Rosebrock, Adrian. "YOLO and Tiny-YOLO object detection on the Raspberry Pi and Movidius NCS." *Pyimagesearch*. January 27, 2020. https://pyimagesearch.com/2020/01/27/yolo-and-tiny-yolo-object-detection-on-the-raspberry-pi-and-movidius-ncs/

[SAE 2021]
SAE. *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*. J3016_202104. April 2021.

[SAE ARP4754b]
SAE. *Guidelines for Development of Civil Aircraft and Systems.* ARP4754b. December 2023. https://www.sae.org/standards/content/arp4754b/

[SAE ARP4761a]
SAE. *Guidelines for Conducting the Safety Assessment Process on Civil Aircraft, Systems, and Equipment*. ARP4761a. December 2023. https://www.sae.org/standards/content/arp4761a/

[Schäfer 2024]
Schäfer, Arndt; Esterbauer, Reinhold & Kubicek, Bettina. *Trusting robots: a relational trust definition based on human intentionality.* Humanities Social Science Communications 11, Article 1412. October 2024. https://doi.org/10.1057/s41599-024-03897-3

[See 2021]

See, Judi E. *Human Readiness Levels Explained.* SAND2021-4299J. Sandia National Laboratories, 2021. https://www.osti.gov/servlets/purl/1787523

[Sharif 2016]

Sharif, Mahmood; Bhagavatula, Sruti; Bauer, Lujo & Reiter, Michael K. *Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition.* In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16). Association for Computing Machinery. 2016. https://doi.org/10.1145/2976749.2978392

[Solingen 1999]

Solingen, Rini & Berghout, Egon. (1999). *The Goal/Question/Metric Method: A Practical Guide for Quality Improvement of Software Development.* The McGraw-Hill Companies. 1999. ISBN 007 709553 7 https://www.researchgate.net/publication/243765439_The_GoalQuestionMetric_Method_A_Practical_Guide_for_Quality_Improvement_of_Software_Development

[Stewart 2015]

Stewart, Katie; Allen, Julia; Valdez, Michelle; & Young, Lisa. *Measuring What Matters Workshop Report*. CMU/SEI-2015-TN-002. Software Engineering Institute. 2015. https://doi.org/10.1184/R1/6575462.v1

[Stumborg 2021]

Stumborg, Michael F; Roh, Becky & Rosen, Mark. *Dimensions of Autonomous Decision-making*. DRM-2021-U-030642-1Rev. Center for Naval Intelligence. 2021. https://www.cna.org/reports/2022/01/Dimensions-of-Autonomous-Decision-making.pdf

[Teare 2025]

Teare, Gené. Startup Funding Regained Its Footing in 2024 as AI Became the Star of the Show. *Crunchbase News*. January 7, 2025. https://news.crunchbase.com/venture/global-funding-data-analysis-ai-eoy-2024/

[Timperley 2023]

Timperley, Chris. Breaking Bots: Robustness Testing for ROS. *RosCON '23, New Orleans.* October 2023. https://roscon.ros.org/2023/talks/Breaking_Bots_Robustness_Testing_for_ROS.pdf

[TRMC 2025]

Test Resource Management Center (TRMC). *TRMC Website*. March 11, 2025. [accessed]. https://www.trmc.osd.mil/

[Trusted-AI 2025]

AI Fairness 360. Trusted-AI. *GitHub*. April 8, 2025. [accessed]. https://github.com/Trusted-AI/AIF360

[Turri 2022]
Turri, Violet & Heim, Eric. Bridging the Gap Between Requirements Engineering and Model Evaluation in Machine Learning [blog post]. *SEI Blog*. December 2022. https://insights.sei.cmu.edu/blog/bridging-the-gap-between-requirements-engineering-and-model-evaluation-in-machine-learning/

[Two Six Technologies 2025]
Two Six Technologies. Armory. *GitHub*. February 7, 2025. [accessed] https://github.com/twosixlabs/armory-library

[UL 4600]
Underwriters Laboratories. *Standard for Evaluation of Autonomous Products*. ANSI/UL 4600. 2020. https://www.shopulstandards.com/ProductDetail.aspx?productid=UL4600

[UMAA 2019]
Unmanned Maritime Autonomy Architecture (UMAA) Architecture Design Description (ADD). Version 1.1a, UMAA-INF-ADD. AUVSI, December 19, 2019. https://www.auvsi.org/sites/default/files/PDFs/UMAA/UMAA%20ADD%20Dec%2019%2C%202019%20v1.1.pdf

[VanHoudnos 2024]
VanHoudnos, Nathan M.; Smith, Carol J.; Churilla, Matt; Lau, Shing-hon; McIlvenny, Lauren; & Touhill, Greg. *Counter AI: What Is It and What Can You Do About It?* Software Engineering Institute, Carnegie Mellon University. October 7, 2024. https://insights.sei.cmu.edu/library/counter-ai-what-is-it-and-what-can-you-do-about-it/

[VDE 2022]
VCIO based description of systems for AI trustworthiness characterization. VDE SPEC 90012 V1.0. VDE 2022. https://www.vde.com/resource/blob/2242194/a24b13db01773747e6b7bba4ce20ea60/vcio-based-description-of-systems-for-ai-trustworthiness-characterisationvde-spec-90012-v1-0--en--data.pdf

[Vincent 2024]
Vincent, Brandi. Why the Pentagon didn't request higher funding for AI in fiscal 2025. *DefenseScoop*. March 11, 2024. https://defensescoop.com/2024/03/11/pentagon-ai-budget-request-2025/

[Wei 2024]
Wei, Dangang. Demystifying Temperature in Machine Learning [blog post]. *Demystifying Machine Learning.* May 9, 2024. https://medium.com/@weidagang/demystifying-temperature-in-machine-learning-ef6828ad4e2d

[White House 2021]
The White House. *Interim National Security Strategic Guidance*. The White House. March 2021. https://www.whitehouse.gov/wp-content/uploads/2021/03/NSC-1v2.pdf

Legal Markings

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

DM25-0496

Contact Us

Software Engineering Institute
4500 Fifth Avenue, Pittsburgh, PA 15213-2612

Phone: 412/268.5800 | 888.201.4479
Web: www.sei.cmu.edu
Email: info@sei.cmu.edu