412TW-TIH-20-02

TEST AND EVALUATION OF AUTONOMY FOR AIR PLATFORMS

RILEY A. LIVERMORE
Captain, USAF

AVERY W. LEONARD
1st Lieutenant, USAF

412TW

JULY 2020

TECHNICAL INFORMATION HANDBOOK

412TH TEST WING
EDWARDS AIR FORCE BASE, CALIFORNIA
AIR FORCE MATERIEL COMMAND
UNITED STATES AIR FORCE

This handbook, 412TW-TIH-20-02, *Test and Evaluation of Autonomy for Air Platforms*, was submitted under job order number CRZ10100 by the Commander, 412th Test Wing, Edwards AFB, California 93524-6843.

Prepared by:

LIVERMORE.RILEY.ALLAN.1368172371 Digitally signed by LIVERMORE.RILEY.ALLAN.1368172371 Date: 2020.07.09 12:47:34 -04'00'

RILEY A. LIVERMORE
Captain, USAF
Project Engineer

LEONARD.AVERY.W.1457527698 Digitally signed by LEONARD.AVERY.W.1457527698 Date: 2020.07.09 09:38:34 -07'00'

AVERY W. LEONARD
1st Lieutenant, USAF
Project Engineer

This handbook has been reviewed and is approved for publication: 9 July 2020

JEROME.ELISABETTA.LIDIA.1265159482 Digitally signed by JEROME.ELISABETTA.LIDIA.1265159482 Date: 2020.07.09 14:54:00 -05'00'

ELISABETTA L. JEROME PhD, SL
Technical Advisor
Air Force Test Center

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)* 09-07-2020	**2. REPORT TYPE** Technical Information Handbook	**3. DATES COVERED** *(From - Through)* July 2019 – April 2020

4. TITLE AND SUBTITLE *Test and Evaluation of Autonomy for Air Platforms*	**5A. CONTRACT NUMBER** SC: 012100
	5B. GRANT NUMBER
	5C. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Riley A. Livermore, Capt, USAF; Avery W. Leonard, 1st Lt, USAF	**5D. PROJECT NUMBER** CRZ10100
	5E. TASK NUMBER
	5F. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) 412 OG/ET CTF, 195 E Popson Ave, Edwards AFB CA 93524	**8. PERFORMING ORGANIZATION REPORT NUMBER** 412TW-TIH-20-02
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Test Center (AFTC), 307 E. Popson Ave, Edwards AFB, CA 93524-6842	**10. SPONSOR/MONITOR'S ACRONYM(S)**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT
DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited. 412TW-PA-20312

13. SUPPLEMENTARY NOTES
CA: 412th Test Wing Edwards AFB CA Print this document in COLOR.

14. ABSTRACT
This handbook provides tools, approaches, and insights to confidently approach the testing of autonomy on air platforms. This handbook outlines what specific considerations for autonomy needs to take place at each phase of the test process to ensure a safe, secure, effective, and efficient test. Two paradigms are presented for the test approach to autonomy. The first focuses on adapting the three phases of test: planning, executing, and analyzing to better test autonomy. The second paradigm leverages the Agile and Development and Operations (DevOps) philosophies to optimize the test and evaluation of autonomy within the larger context of autonomy development and fielding.

15. SUBJECT TERMS
Agile; Air/Aerial Systems; Algorithm; Autonomy; DevOps(Development and Operations); Flight Test; LVC(Live-Virtual-Constructive); OSA(Open Systems Architecture); RTA(Run Time Assurance); Service; Surrogate Platform; T&E(Test and Evaluation); Task; UAS(Unmanned Aerial System/Systems)

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19A. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Jeffrey Jessen, Chief Engineer Emerging Technologies CTF
U	U	U	Same as Report	54	**19B. TELEPHONE NUMBER (INCLUDE AREA CODE)** (661) 275-5979

STANDARD FORM 298 (REV. 8-98)
PRESCRIBED BY ANSI STD. Z39.18

EXECUTIVE SUMMARY

The 2018 National Defense Strategy emphasizes that the effective implementation of autonomy is essential for future engagements. Key to this implementation is the ability to test and evaluate systems that perform autonomous tasks. The purpose of this handbook is to equip testers with tools, approaches, and insights to confidently approach the testing of autonomy on air platforms. The air domain is chosen specifically for its applicability to the Air Force mission and to help scope the focus of this handbook. The intent is not to be an exhaustive reference for testing and evaluating autonomy; rather, the goal of this handbook is to provide a launching point for greater investigation.

This handbook begins by outlining the purpose, key definitions, and foundational assumptions to help clarify what is meant by "autonomy" for the Air Force Test Center. After establishing this foundational understanding, the test approach for autonomy is presented. Testing autonomy on airborne platforms brings unique challenges and, therefore, to be successful testing should leverage the five principles of: Early User Involvement, Continuous and Cumulative Feedback, Streamlined Process and Products, Pilot Training Approach, and Human-Machine Interaction Consideration. The overarching test approach is split into two paradigms: three phases of test and Agile Development and Operations (DevOps).

First, the current flight test paradigm of plan, execute, and analyze (three phases) is reexamined through the lens of testing autonomy. Of the three phases, executing a test of autonomy is currently the biggest challenge within the test community. The goal of any test, and of autonomy in particular, is that it is safe, secure, effective, and efficient. Therefore, the optimal test execution of an autonomous task or mission should include some type of run time assurance, live-virtual-constructive, open systems architecture, and a surrogate platform. To support the execution of autonomy testing that is responsive and timely, a close pairing of the planning and analysis phases is recommended. Different approaches, like design of experiments, human factors considerations, hardware-in-the-loop, and systems theoretic process analysis, are provided as tools for facilitating the needed planning and analysis capabilities for autonomy test. Within this paradigm, the goal is to adapt existing practices and modify to facilitate a spiral test and evaluation strategy of autonomy.

The second paradigm features a more dramatic shift from the current test practices to one that incorporates the principles of Agile and DevOps. The Agile philosophy emphasizes individuals and interactions over processes and tools, working products over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan. It has been instrumental in modern software development and will certainly play a role going forward with the development of autonomy. Additionally, DevOps is the revolutionary concept that integrates the development and operation functions of an organization. This process relies on continuous integration, continuous delivery, continuous monitoring and logging, microservices, and collaborative, cross-functional teams to implement high-quality products, quickly. As the Air Force begins to adopt more of the Agile and DevOps principles, it will be able to realize the DoD's goals of "delivering at the speed of relevance," "organize for innovation," and "streamline rapid, iterative approaches from development to fielding."

This page intentionally left blank.

TABLE OF CONTENTS

This page intentionally left blank.

INTRODUCTION

In an effort to maximize warfighter lethality, autonomy must be incorporated into our systems, but first it must be developed, tested, and fielded. The 2018 Summary of the National Defense Strategy of the United States of America (reference 1) and the 2019 National Defense Authorization Act (reference 2) have made it expressly clear that fielding systems that incorporate autonomy is essential for the United States to retain its competitive edge. Furthermore, both of these documents have called for reform of the DoD acquisition process to produce results “at the speed of relevance”. In 2019, the Air Force Chief Scientist released “Autonomous Horizons: The Way Forward” (reference 3) that built a roadmap and framework for advancing the state of the art in autonomy, while supporting its transition to existing systems. In response to this higher-level guidance, the Air Force Test Center (AFTC) has crafted a strategic goal to “shape future test and evaluation capabilities to maximize warfighter lethality.” A key tenet of this strategic goal is to build a capability to test systems that feature autonomy and machine learning algorithms. This handbook supports the AFTC strategic goal and provides a reference for the considerations and approaches which should be taken for the testing of autonomy on airborne platforms.

This handbook is organized as follows: first, the specific purpose of the handbook, key terms, and foundational assumptions are defined. Additionally, the first section outlines the scope of the handbook. The next section develops the test approach for testing autonomy on airborne platforms. This section begins by detailing how the current test paradigm should be adapted to test autonomy. It then discusses how a new test paradigm, using the Agile and Development and Operations (DevOps) principles, should be implemented to maximize the effectiveness of the autonomy. Finally, the high points of testing autonomy are summarized and the path forward is highlighted. Additionally, appendices which include details about current autonomy test are included, as well as some other helpful references.

PURPOSE

The purpose of this handbook is to equip testers with tools, approaches, and insights to confidently test autonomy on air platforms. Due to the breadth of challenges with testing autonomy and the overall immaturity of existing test capabilities, the content of this handbook does not explicitly define specific best practices. Rather, this handbook outlines the considerations for autonomy test, which need to take place at each phase of the test process to ensure a safe, secure, effective, and efficient test. This handbook is intended to be more of a compass, than a map.

This handbook assumes that the reader has some flight test experience and/or a technical background. Test and evaluation of autonomy is fundamentally a multi-disciplinary venture; therefore, this handbook addresses some of the diverse perspectives required to be successful. The intent is not to be an exhaustive reference for testing and evaluating autonomy; rather, the goal of this handbook is to provide a launching point for greater investigation. There exist numerous studies, performed throughout the DoD, and a large body of academic research that this handbook leverages and seeks to build upon. Furthermore, this handbook acknowledges that the testing of autonomy cannot be considered in a vacuum; it proposes concepts and frameworks that developers and acquirers of autonomous systems need to best integrate with the test community. The ultimate success of autonomy test hinges on future program offices heeding this handbook and appropriately planning for test early on in the program.

KEY DEFINITIONS

Autonomy: a set of intelligence-based capabilities that can respond to situations that were not pre-programmed or anticipated prior to deployment, as defined in *Autonomy Community of Interest (COI) Test and Evaluation* (reference 4). Autonomy constitutes a degree of self-sufficiency and self-directed behavior (with human's proxy for decisions) as defined in *The Seven Deadly Myths of 'Autonomous Systems'* (reference 5).

Autonomy Engine: hardware onboard a platform or system which hosts autonomy software.

Complex System: a system in which understanding the interactions between parts is equally as important as understanding the individual part's function, as defined in *Dynamics of Complex Systems* (reference 6).

Service: a stand-alone, encapsulated unit of functionality that receives inputs and gives outputs via a defined interface (primarily for software) as defined in *Service-oriented Modelling* (reference 7).

Task: composed of one or more services that accomplishes a specific objective or action over a discrete, defined period of time.

Trust: the willingness of a party to be vulnerable to the actions of another party based on the expectation that the other will perform a particular action important to the trustor, irrespective of the ability to monitor or control that party, as defined in *An Integrative Model of Organizational Trust* (reference 8). Trust is built primarily on the satisfactorily demonstrated combination of reliability and accuracy.

FOUNDATIONAL ASSUMPTIONS

Even though autonomy was defined in the previous section, it is important to further clarify what "autonomy" practically means for the purposes of this handbook. The following four foundational assumptions are primarily derived from the 2012 Defense Science Board study on the "Role of Autonomy in DoD Systems" (reference 9) and help hone the meaning of "autonomy":

1. "Autonomy" is fundamentally software-based.
2. "Autonomy" is considered relative to specific tasks, over a discrete time interval, and not for entire systems.
3. A "Fully Autonomous" system is a misnomer. All autonomy must interact with a human at some point and is therefore considered a joint human-machine cognitive system.
4. Determining and arguing about "Levels of Autonomy" is unhelpful and counter-productive to testing and evaluating the system.

HANDBOOK SCOPE

This handbook focuses on testing and evaluating autonomy for the air domain. The air domain is chosen because of its pertinence to the Air Force's mission and because of the unique challenges it brings to the T&E enterprise that are not as applicable to other domains. While the examples presented are for airborne autonomy applications, the majority of the frameworks and tools presented in this handbook are generally applicable to the other domains.

For this handbook, the types of autonomy considered as the system under test (SUT) are ones that feature real-time applications in the physical world, often referred to as "autonomy in motion" as in *Defense Science Board: Summer Study on Autonomy* (reference 10). This is opposed to "autonomy at rest" applications, which are essential for building a more lethal force and will play a vital role in testing and fielding “autonomy in motion” systems. Examples of “autonomy in motion” could be found in a smart bomb, an unmanned aerial system (UAS), or even as a subcomponent onboard a manned platform (i.e., the ground collision avoidance system onboard the F-16). A helpful method for determining if a system constitutes “autonomy in motion” is to think in terms of John Boyd’s “OODA” loop from *A Discourse on Winning and Losing* (reference 11). Autonomy in motion must:

- **O**bserve its surroundings to create a worldview,
- **O**rient itself within the worldview,
- **D**ecide on an appropriate response,
- **A**ct on that decision.

Figure 1 gives a hierarchical perspective that illustrates the role of autonomy for one platform. The missions that the platform executes are composed of a variety of tasks, which are built on lower-level services. When the word “autonomy” is used in this handbook, it is referring to either the actions taken at the task or at the mission level, which are enabled by the services. In most contexts, these actions are currently being performed by human operators. Traditionally, the test community does not test or evaluate the human operator and therefore the test paradigm must be expanded to incorporate human operator-like evaluation criteria for an autonomous task or mission.

To better understand figure 1, the Suppression of Enemy Aerial Defense (SEAD) mission is used as an example. For a fighter platform performing SEAD, jamming the enemy radar is one of the primary tasks. Additionally, tasks like airspace integration, aerial refueling, and formation flying might all be required to successfully accomplish the mission. These tasks, in turn, will require a host of services that are responsible for operating the various payloads, the aircraft itself, and the command and control functionality.

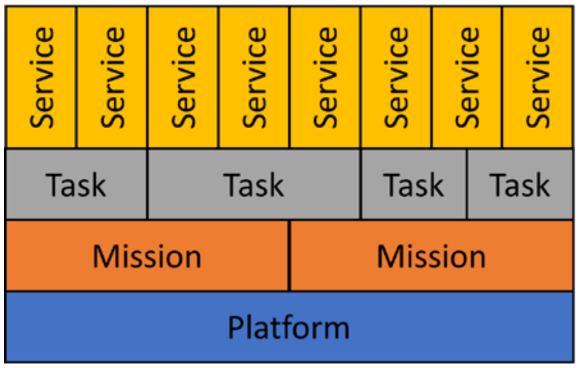

Figure 1 Mission Hierarchy for a Platform

The words machine learning (ML) and artificial intelligence (AI) were intentionally avoided in this handbook. ML and AI are tools which enable autonomy; strategies presented in this handbook apply to the test of autonomous systems regardless of how the autonomy was created or how it functions.

TEST APPROACH

With the emergence of new technologies and the changing nature of the world, revolutionary approaches must be embraced to stay relevant, credible, and timely. This section presents the path forward to success by splitting the test approach into two phases. The first phase features an evolution of the current test paradigm. The three phases of test (planning, execution, and analysis) are investigated through the lens of testing autonomy. Next, the paradigm shift towards an Agile DevOps approach to autonomy test is discussed. In this new paradigm, the traditional, siloed approaches of development, test, and operations are removed, and a vision for a truly integrated ecosystem of development, test, and operations is presented. These two test paradigms should be viewed as complimentary in nature and not antagonistic. To fully achieve the organizational approach of Agile DevOps, the evolutionary progress must first be made. From a practical perspective, the short-term emphasis will predominantly be on the first paradigm, with the long term goal of realizing the second paradigm.

When considering testing of autonomy, there are several overarching principles which are essential. These principles are key to "delivering combat capability at the speed of relevance" (reference 1).

1. **Early User Involvement**: To adequately test and evaluate autonomous tasks and missions, the users must be included as early as possible. This means that the traditionally distinct developmental test (DT) and operational test (OT) must be holistically integrated. Obtaining operational feedback early is pivotal to both the requirements generation process as well as the testing of autonomy.
2. **Continuous and Cumulative Feedback**: The principle of "fail fast" is important when testing autonomy. Feedback on performance should be provided as soon as possible to better inform the development and improvements of the autonomy.
3. **Streamline Processes and Products**: To enable fielding at the "speed of relevance", the test team must be empowered to tailor test processes to adequately match the requirements of the program. Furthermore, the test team must also have the flexibility to create and design new processes that best meet the program's goals.
4. **Pilot Training Approach:** Perhaps the most unique facet of autonomy test is that oftentimes the autonomy being tested is intended to replicate pilot-like functionality. Traditionally, AFTC has assumed that the operator is sufficiently trained, and therefore is accustomed to testing the system and not the pilot. Testers will have to leverage and adapt training approaches for manned operators to best evaluate the performance of autonomous tasks and missions.
5. **Human Machine Interaction Consideration:** The autonomous task or mission must be viewed within the larger context of the human interaction. Human factors elements such as trust, human-autonomy interface, and situational awareness are crucial to the ultimate success of an autonomous task or mission.

With the emergence of ever-new autonomous capabilities, these principles must be "baked-into" each program from the beginning. This will help minimize any roadblocks caused by the cultural shift away from the traditional acquisitions and test approach. Furthermore, the testing of autonomy should not end when the autonomy is fielded. End users must be empowered to provide feedback to the developers. The test team can bridge this gap by soliciting feedback from operational units and through observing real-world data.

PARADIGM EVOLUTION: 3 PHASES OF TEST

The Test Resource Management Center (TRMC)[1] has strategically divided the task of testing and evaluating autonomy into three categories: test planning, test execution and control, and performance assessment, and analysis, as found in *Assessment of Test and Evaluation Infrastructure and Improvements Required for Future Emerging Technologies* (reference 12). These three categories map nicely to the general flight test process, and therefore provide a perfect launching point for what evolutionary steps are needed to best facilitate T&E of autonomy.

For flight test in general, the **safety**, **security**, **effectiveness,** and **efficiency** considerations are the primary motivators for how the test is conducted. These four categories help shape all aspects of the test process and are the primary motivators for the testing of autonomy as well. Oftentimes, the safety and security considerations are in opposition to the test effectiveness and efficiency. The tension between these four facets is even further exacerbated when testing autonomy specifically due to the inherent complexities, interdependencies, and increased uncertainty of outcome. Accounting for these factors must take place in all three phases of test.

The most near-term obstacle in the world of autonomy test centers on execution. The ability to integrate and fly an autonomous task on an air platform on a military range is a significant challenge, due to the relative infancy of autonomy test when compared to traditional test. Therefore, the best way forward to execute an autonomy test is discussed first. Once the infrastructure is in place to test autonomy, the larger questions of how to properly plan and analyze the test can be answered. While the planning and analysis phases serve as bookends in a traditional test program, for autonomy they must be more tightly coupled to be effective and responsive. The considerations required for test planning and analysis of autonomy are discussed in light of how the autonomy test should be executed.

EXECUTION

Currently, of the three phases of test, the biggest challenge for the T&E community is in executing an autonomy test. Safety tends to be the driving concern, but there are also issues with system integration, platform availability, accurately stimulating the autonomy in real-time, and other logistical issues of range space, communication networks, etc. To execute an autonomy test that is safe, secure, effective, and efficient, one approach is to use a test-specific middleware that facilitates four components: Run Time Assurance (RTA), Modeling & Simulation (M&S) to include Live-Virtual-Constructive (LVC), Open Systems Architecture (OSA), and implementation on surrogate test platforms. These four components are represented as a Venn diagram in figure 2. Each component provides a crucial function in the testing of autonomy, while also being a key enabler for other components. While it is possible to test autonomy with fewer components, this construct optimizes the balance of safety, security, effectiveness, and efficiency.

In the proceeding subsections, the rationale for choosing each component as well as the unique impact that each component provides to the safe testing of autonomy is discussed. It is important to note that while the safety, security, effectiveness, and efficiency gains made by each component are independently discussed, they overlap and enable each other. The result is that this four-component framework produces a guide to executing an autonomy test that is greater than the sum of its individual parts[2].

[1] TRMC's Autonomy and Artificial Intelligence Testing (AAIT) program area has compiled a database of tools, programs, and gaps for autonomy testing. They also provide government-owned software tools for planning, executing, and analyzing autonomy. Contact Vernon Panei (Vernon.panei@navy.mil) for more information and for permission to access the database.

[2]This framework is more than theoretical, Appendix B contains two examples of current systems used for autonomy test that feature these four components. A system description, details of flight testing, and future development efforts for both the Testing of Autonomy in Complex Environments (TACE) and Cooperative Operations in Denied Environments (CODE) systems are included.

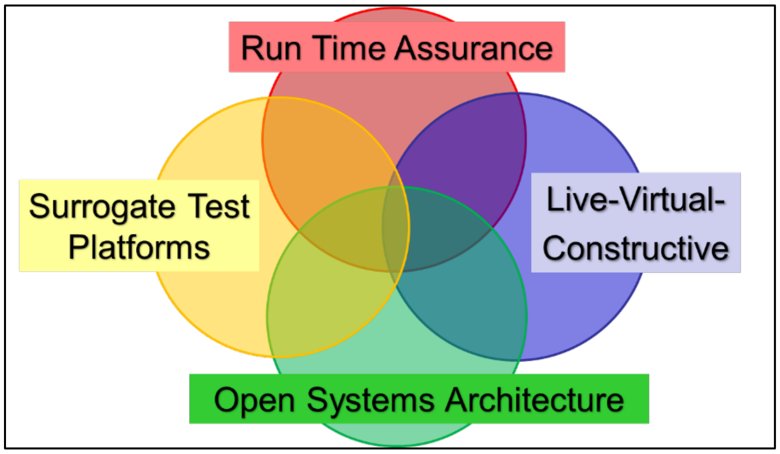

Figure 2 Four Components for Autonomy Test Execution

Run Time Assurance (RTA):

The first assumption of this handbook states that autonomy is fundamentally software-based. Therefore, a critical component of autonomy test is ensuring that the software running on the autonomy engine can be tested in a way that is safe, secure, effective, and efficient. Due to the complexities associated with autonomy, utilizing traditional, exhaustive, offline test approaches are intractable to verify that the autonomy will not exhibit emergent or undesirable behavior. One popular approach to this unique challenge is using RTA algorithms to provide system reliability in real time. RTA can be summarized as a deterministic wrapper that serves two primary functions: reliably detect problems (software or environmental) and switch to a recovery/safe mode in the event of a failure as described in *A Study on Run Time Assurance for Complex Cyber Physical System* (reference 13).

The two functions of a typical RTA system are shown in figure 3. The unverified "Advanced Controller" is analogous to the autonomy under test. If any problem occurs with the "Advanced Controller" during execution, then the RTA switches to the "Recovery Controller." This framework enables test of any algorithm that meets its interface requirements, regardless of complexity. Furthermore, this basic architecture in figure 3 can be easily extended to run multiple RTA algorithms in serial or incorporate multiple recovery controllers to safely enable full performance envelope recovery, as described in "Run Time Assurance for Complex Autonomy" (reference 14). Additionally, there can be RTA algorithms incorporated within the autonomy itself, as well as test-specific RTA algorithms that sit outside the autonomy under test. The exact application of the RTA should be tailored to meet the specific test objectives of the autonomous task or mission. The RTA has wide-spread applicability for autonomy test and has been used for vehicles operating in various domains to include ground ("A case study on runtime monitoring of an autonomous research vehicle (ARV) system," reference 15), air ("Nonlinear Adaptive Control of Quadrotor UAVs with Run-Time Safety Assurance," reference 16), and space ("Formally Verified Run Time Assurance Architecture of a 6U CubeSat Attitude Control System," reference 17).

The implementation of RTA is not without its challenges. As the scenarios and underlying services feeding the autonomous tasks become more complicated the RTA's recovery controller must also become more robust. Determining the multi-dimensional, hypersurface safety boundary and interrogating it in real time is a significant challenge. Furthermore, the distinction between "safe" and "unsafe" is often blurry and hard to precisely determine a priori. Despite some of the challenges involved in developing a capable RTA, the use of RTA undoubtedly plays a pivotal role in the safe, secure, effective, and efficient testing of autonomy.

RTA's Role in Test Safety.

Safety is the primary motivation for using RTA. Testing autonomy is unique in that the behaviors are not explicitly known ahead of time. Typically, an autonomous task is given an objective and then chooses the best course of action based on its perceived worldview. Thus, when the autonomy behaves unexpectedly, it does not mean that behavior is "wrong". Therefore, the balance between testing the autonomous behavior and ensuring test safety must be judiciously determined.

A certified RTA system strikes this balance perfectly because it simultaneously allows unverified and unknown autonomous algorithms to execute while guaranteeing that certain safety of flight parameters will always be observed. This effectively creates a "sandbox" in which the autonomy is allowed to operate. The Air Force Test Pilot School has successfully used this similar concept for autonomy testing by using Calspan's Variable Stability Simulator (VSS) Learjet, as documented in "A Limited Evaluation of an Automatic Ground Collision Avoidance System for Performance Limited Aircraft" (reference 18); "Operate Remote Aircraft Clairvoyantly in a Limited Evaluation (ORACLE)" (reference 19); and the Variable In-flight Simulator Test Aircraft (VISTA) F-16 "Flight Test Evaluation of the HAVE RAIDER II Autonomous Wingman Architecture" (reference 20); "Flight Test Evaluation of the AFRL Automated Wingman Functions" (reference 21) as manned autonomy surrogates. The use of RTA in both the VISTA and VSS provides an envelope protection systems that prevents the aircraft from departing controlled flight while flying the autonomous task.

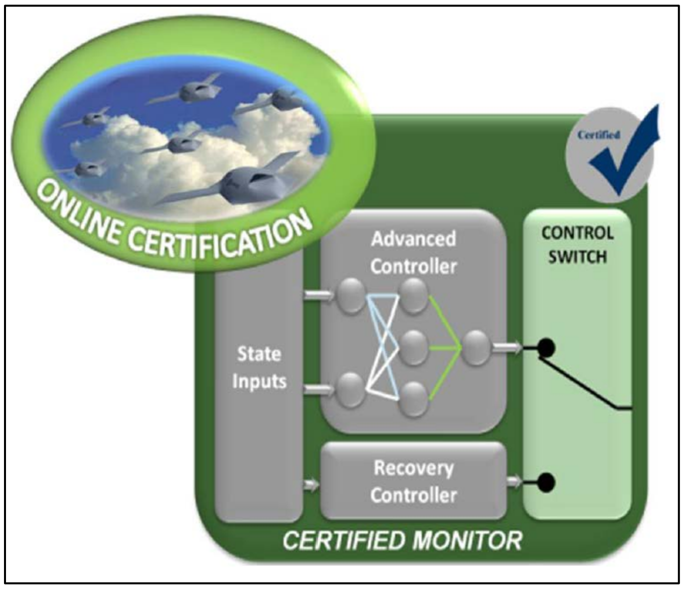

Figure 3 Run Time Assurance of Unverified Flight Critical Software (reference 13)

RTA's Role in Test Security.

Security is crucial when it comes to testing. One of the concerns that commonly arises with unmanned platforms is a cyber-attack against the autonomy under test. This attack could be the result of malicious code within the autonomy or could come from external influences such as jamming or spoofing. The RTA plays a role in test security because it adds another layer of defense against an adversarial action, by providing a counter to corrupted code, verifying commands from the ground, and providing redundant control.

RTA's Role in Test Effectiveness.

The RTA also provides an effective means for testing autonomy because it allows for more aggressive testing of autonomy. When testing a complex system, the process of determining where the system works is sometimes best accomplished by discovering where it fails. Often, this "failure" is a result of failing to manage the quantity or quality of the information shared among numerous services. Using a trusted RTA allows for a more aggressive test approach because of the reliability provided by the RTA. The ability to pursue an aggressive approach helps the test team effectively provide feedback on the autonomy's performance and rapidly identify areas of deficiency.

RTA's Role in Test Efficiency.

The RTA construct promotes efficient testing as well because it allows new software updates and algorithms to be rapidly tested. With the rise of agile software development practices, the emphasis has shifted to evolutionary development, early delivery, and continuous testing as described in *Manifesto for Agile Software Development* (reference 22). The rapid pace of development of autonomous algorithms is hindered by the inadequacy of current test processes to test these algorithms in an appropriate time. Leveraging RTA provides a streamlined route to test updates in software that is responsive to the pace of development.

Live-Virtual-Constructive (LVC):

Leveraging live, virtual, and constructive entities is invaluable to Autonomy test. LVC, a form of M&S, is a concept that is becoming more widespread in the engineering and test vernacular. The concept allows for the fusion of real-world test assets (live), human-operated, simulated entities (virtual), and computer-controlled, simulated entities (constructive) to interact seamlessly with each other as described in *Modeling and Simulation (M&S) Glossary* (reference 23). Properly implementing the virtual and constructive entities in the live scenario is incumbent on a robust M&S capability. The importance of LVC is two-fold in testing of autonomy; firstly, it is invaluable in testing the autonomous tasks themselves, and secondly, it is crucial in evaluating the link between the human and the machine.

LVC's Role in Test Safety.

The LVC capabilities are crucial for safely testing autonomy because they leverage virtual and constructive effects to test the onboard autonomy while preventing the live aircraft from entering into a dangerous scenario. For example, safe testing of an airborne collision-avoidance task can be performed using a live aircraft flying towards a constructive aircraft. Since the world view of autonomy onboard the live aircraft is manipulated by the LVC environment, the constructive aircraft appears "real" to the live aircraft. Therefore, the autonomy onboard the live aircraft behaves identically as it would if the approaching aircraft was actually a live asset. Therefore the autonomy software, running onboard a live aircraft, is fully tested while simultaneously mitigating the risk of a mid-air collision. The hardware sensors, which the autonomy would use to detect the live aircraft, could be tested separately.

LVC is also valuable in safely evaluating the trustworthiness of the autonomous task from a human perspective. The loyal wingman use case is a salient example that highlights the value of using LVC for evaluating the manned-unmanned interaction. Loyal wingman features an unmanned wingman acting as a force multiplier for a manned aircraft. The obvious safety concern with this scenario is that an unmanned aircraft is flying in close proximity with a manned aircraft. As part of a build-up approach, the algorithm could first be tested with the pilot flying in a simulator (virtual aircraft) while the live autonomous wingman flies in formation. Utilizing LVC capabilities in this scenario not only ensures a safer test but also gives the human operator a chance to interact with the autonomy and provide constructive feedback.

LVC's Role in Test Security.

LVC can play a crucial role in bolstering test security by reducing the amount of live assets and sensors needed to conduct a test. When trying to determine the performance of an autonomous task in an operationally relevant environment, it might be prudent to use LVC to negate potential security concerns. Oftentimes, the combination of multiple systems working together, emitting certain frequencies, using certain sensors or radios in a live environment might cue adversaries onto current capabilities or what the end goal of the technology may be. Using LVC can obfuscate the roles of the live test assets from their operational mission context by simulating other entities, systems, and effects that would impact the security classification. Additionally, LVC can bring high-fidelity red team capabilities to bear in a manner that would be more difficult to compromise.

LVC's Role in Test Effectiveness.

The effectiveness of autonomy testing is magnified through the incorporation of virtual and constructive entities. Oftentimes, there are specific flight and environmental conditions that are difficult, costly, or impractical to achieve (e.g., GPS denial, sensor degradation, severe weather affects, multi-ship operations, etc.). Properly implementing LVC allows for the proper execution of these variables at specific test points. This is especially pertinent when testing edge cases in the autonomy logic. When trying to create a scenario where the autonomy might fail, the ability to augment the live environment is invaluable. Using the same air collision avoidance scenario as before, the test team could identify the effectiveness of the algorithm with degraded sensors. In real life, this would be risky, difficult, and costly to accomplish, but with LVC it is straightforward. Another example is testing how the air collision avoidance task performs in an environment with multiple aircraft. Setting up this scenario without LVC would require increased coordination, assets, personnel, and planning. With LVC, adding additional aircraft is only constrained by the hardware limitations of the simulation computers and of the network.

LVC's Role in Test Efficiency.

Lastly, there are tremendous efficiencies to be gained by leveraging LVC. The future testing of autonomy will undoubtedly feature a blend of both live test and simulation. To be both responsive to change and cost-effective, the majority of the test points will have to be executed in simulation, with only the salient test points being flown in real life.

An accurate LVC environment is the key to ensuring that the simulation capabilities are compatible with the live flight test data. Once this accuracy is assured, it opens the door for a number of efficiencies. A faster-than-real-time simulation capability is one of these key efficiencies that enables a large number of test points to be executed in a small amount of time. Additionally, LVC is an integral part of hardware in-the-loop (HIL) simulations. Having a framework to do HIL simulation boosts efficiency by bridging the gap between the simulation and live test environments. This capability provides the ability to simulate the autonomy in a way that will be analogous to live flight. This boosts efficiency by providing an environment that supports an agile software development approach, allowing for new capabilities to be rapidly developed and tested. Most importantly, this tool is critical in maximizing the likelihood of success in live flight test.

As mentioned earlier, LVC provides various entry points for human interaction with the autonomy along the various stages of development. This access promotes more efficient testing because it allows for human feedback earlier in the development process. The early integration of feedback on human-autonomy interactions bolsters efficiency by providing constant vector checks to the developers. Using LVC throughout the autonomy T&E process will ultimately reduce wasted time because of the ability to provide timely feedback on the performance of the autonomy.

Open Systems Architecture (OSA):

In the past, when procuring airborne weapon systems, the government has purchased end-to-end solutions from prime integrators. While initially convenient, the proprietary nature of the system's operational flight program makes it difficult and expensive to adapt to changing demands for the system. Recently, all three Service Secretaries have emphasized the vitality of using OSAs for future acquisition activities as documented in "Modular Open Systems Approaches for our Weapon Systems is a Warfighting Imperative" (reference 24). The DoD generally defines an OSA as "a structure in which system interfaces share common, widely accepted standards, with which conformance can be verified" as found in "Modular Open Systems Approach" (reference 25). There is a widespread understanding throughout the DoD that OSAs are an integral part in operating at the "speed of relevance" (reference 1).

This trend towards OSAs is even more important when considering the testing of autonomy. The Air Force Research Laboratory (AFRL) developed the Unmanned Systems Autonomy Services (UxAS), as described in "A Brief Introduction to Unmanned Systems Autonomy Services (UxAS)," (reference 26) and the Github Repository (reference 27) to provide a modular, extensible, services-based framework for testing autonomy. Additionally, AFRL is currently working on the Teaming-Enabled Architectures for Manned-Unmanned Systems (TEAMS) which is an iterative, model-based approach that will enable the rapid testing of future manned-unmanned teaming technologies, as described in "U.S. Air Force Research Lab Awards GE TEAMS Program" (reference 28). More generally, the Open Missions Systems (OMS) architecture is gaining more widespread traction throughout the DoD. OMS is a non-proprietary, open architecture standard for integrating subsystems and services into mission packages. The challenge of OSAs like OMS is that they are easily extensible to include a variety of functionality, but require a critical abstraction layer (CAL) to be implemented on a particular platform. Therefore, a particular autonomy could be compliant with the desired OSA, but still not work on the particular platform because of incompatibility with the CAL. This nuance highlights the larger challenge with OSAs, which aim to be broadly applicable to a wide array of functionality, while maintaining practical functionality and interoperability. Regardless of the specific architecture used, OSAs are a key enabler that directly supports the other three key components of the autonomy test infrastructure.

It is important to note that intellectual property can still be maintained with an OSA. The OSA specifies the interfaces and formats that allow the data transfer to occur, but don't require full software transparency. This is important because it protects the intellectual property of the developers, while simultaneously allowing for a more modular and interoperable approach. Using an OSA prevents the development of monolithic autonomy algorithms and instead steers developers to produce software geared towards specific capabilities, also known as services. The advantage of using an OSA that supports the use of services is that each algorithm is logically represented as a self-contained activity. Furthermore, orientation towards services facilitates a services-based testing of autonomy (SBTA) approach. This is an effective way to explicitly isolate, test, and evaluate the specific autonomous task of interest, as described in "Services-Based Testing of Autonomy (SBTA)" (reference 29). Using a SBTA approach in conjunction with an OSA has tremendous benefits for the safety, effectiveness, and efficiency of autonomy test.

OSA's Role in Test Safety.

The primary safety benefit of using an OSA is that it allows for the easy integration of the autonomy engine with the RTA. Once the RTA is determined to be sufficient for flight testing, that configuration can be used with any autonomy engine and platform that is compliant with its services-based interface. This guarantees the reusability of the RTA and reduces unnecessary regression testing.

Additionally, using an OSA to execute a SBTA approach improves test safety by allowing for the test team to isolate and test specific services within an autonomy task. This specificity is essential for safely testing complex systems comprised of numerous services. Honing in on a specific service increases the clarity on what exactly is being tested and what the expected outcomes should be. This also aids in diagnosing the root causes of problems with the autonomous task, because the various services can be isolated and tested independently.

OSA's Role in Test Security.

There is a misconception that because software is considered "open" (i.e., access to source code or use of using publicly available software) it is inherently less secure. The irony with this misconception is that open source software is more secure precisely because the code is available. This is largely because the wide access helps identify vulnerabilities, which are quickly identified and mitigated. Even the National Security Agency, motivated by this same logic, has published a number of their encryption algorithms for securing Top Secret data as described in "Commercial National Security Algorithm (CNSA) Suite Profile of Certificate Management over CMS" (reference 30).

Leveraging an OSA in the testing of autonomy provides greater test security from potential cyber threats and simultaneously allows for a more rigorous testing of cyber hardness. Firstly, an OSA allows for security threats to be identified and updates to be incorporated relatively seamlessly. Secondly, by using a services-oriented architecture, certain services can be highlighted and tested specifically for cyber vulnerability.

OSA's Role in Test Effectiveness.

Using an OSA contributes to effective testing of autonomy in three specific ways. Firstly, it allows for specific testing of autonomy services. Secondly, it effectively allows for the decoupling of the autonomy from a specific platform. Finally, utilizing an OSA provides a process for easily testing competing autonomous tasks.

Cooperative autonomous search is a good example for clarifying how a services-based approach improves autonomy test effectiveness. The cooperative autonomous search task could be notionally separated into four subcomponents: search, target identification, track, and coordination with participating vehicles. If a comprehensive autonomy was designed to facilitate this task, testing the autonomy would require a run through the full scenario. Additionally, if and when the autonomy performed poorly, it would be harder to diagnose the root cause of the failure.

Conversely, if each of these subcomponents was written as an independent service, they could be specifically tested and evaluated independently. The integration of each of these subcomponents is theoretically straightforward because the OSA provides a ready framework for integration. Also, data logging of message traffic between services in an OSA is a powerful tool for analysis. The performance of the autonomy can be better characterized and analyzed through monitoring the information being passed between the various services during the execution of its autonomous task. This capability provides the framework for performing "white box" analysis, which helps answer the question, "why is the autonomy behaving that way?"

The second measure of effectiveness gained by using an OSA is the effective decoupling of the autonomy from a specific platform. As long as the platform's autopilot can interface with the OSA, then any autonomous task that is similarly compatible should work. This portability allows for the use of autonomy test surrogates. This is crucial for test effectiveness because it increases the number of potential test aircraft. It also gives the test team greater flexibility to tailor the autonomy under test to the right aircraft in order to capture the correct data.

Finally, an important boost in test effectiveness is gained by using an OSA to test competing autonomous tasks. Having an OSA with a defined Application Program Interface (API) enables different vendors to develop autonomous tasks and integrate them onto any vehicle with the same OSA. The result of being able to effectively test and evaluate at the granularity of services enables the government to acquire the most capable service that best aligns with the program's needs.

OSA's Role in Test Efficiency.

Following a SBTA approach in conjunction with an OSA allows for the sharing of services amongst numerous autonomous tasks. For example, the service for coordination between friendly vehicles could be used not only in the cooperative search algorithm, but also in a swarming or an autonomous aerial refueling task. The potential commonality of these services amongst numerous autonomous tasks drastically reduces the test time required. This is analogous to the object-oriented programming methodology that is ubiquitous in the software development world.

Additional efficiencies can be gained by using the OSA's defined interface to streamline the integration of 3rd party developed autonomy. The integration effort to get software to properly work on hardware is not trivial. The integration challenges are exacerbated when multiple, independent entities are trying to get their respective pieces to fit together. Having a well-defined API for the OSA mitigates much of the risk and greatly improves the likelihood of timely integration and test.

The platform-agnostic advantage gained by utilizing an OSA allows for parallel testing of an autonomy across numerous platforms. For example, an immature autonomous task can be first flown on a small, group 1 UAS[3] to get a quick-look at performance, as described in "Unmanned Aircraft System Airspace Integration Plan" (reference 31). As the autonomy matures, it can easily migrate to more advanced platforms because each of the test platforms are OSA-compliant. Additionally, using an LVC configuration allows for virtual and constructive entities to normalize the test environment to ensure continuity between the various platforms.

Lastly, having a defined OSA allows the development environment to mimic both the simulation environment and the live test environment. Maintaining a common OSA for the development, simulation, and flight test allows for rapid integration and testing. This continuity across these different domains and air platforms emphasizes the efficiencies gained through an OSA.

Surrogate Test Platforms:

The capstone in executing a flight test of autonomy is flying a live aircraft running the autonomous task or mission. The compilation of the RTA, LVC, and OSA onboard a physical aircraft presents the culmination of the four components working together. Both a physical aircraft and the appropriate airspace and clearances to operate that aircraft are essential for testing airborne autonomy. The use of various-sized surrogate UAS testbeds has significant implications on the safety, security, effectiveness, and efficiency of the test.

Surrogate Test Platform's Role in Test Safety.

Leveraging a fleet of surrogate testbeds for autonomy test is crucial for test safety, because it allows the test team to choose the size and performance characteristics that match the level of risk that they are willing to assume. For example, a maneuver that is considered high risk for a group 5 UAS could be instead performed on a smaller aircraft, minimizing the consequence of the risk. This allows for a build-up approach where the autonomy can be flown on smaller or slower UAS prior to being flown on larger or faster UAS.

[3] See appendix C for definitions of UAS groups.

This risk reduction better facilitates the testing of autonomy with no detrimental impact to the objectives of the test plan.

<u>Surrogate Test Platform's Role in Test Security.</u>

Oftentimes, the security classification increases when a specific mission is matched with a certain airframe. This is intuitive from a strategic perspective, but can be seriously limiting for a test team. Leveraging surrogate test aircraft that have representative dynamics, radar cross sections, or payload capacity would potentially allow for testing the onboard autonomy at a lower classification level. Regardless, decoupling the autonomy engine from the target platform helps minimize the security risk. One primary way this happens is by abstracting away the specific performance parameters of the target aircraft and instead using relevant numbers for the surrogate aircraft. This impacts not only test execution, but also dampens the burden on test planning and analysis.

<u>Surrogate Test Platform's Role in Test Effectiveness.</u>

Utilizing a fleet of autonomy test surrogate aircraft of various sizes improves test effectiveness because it enables the matching of the autonomy under test with the appropriate aircraft. For less mature autonomy algorithms, flying on a smaller aircraft will provide that immediate feedback that can inform future development. For algorithms that are nearing operational maturity, demonstrations on operational scale vehicles are an essential step. It is important to note that the progression from small to large scale aircraft does not have to be sequential or linear. Instead, the rapid and spiral development cycles present in the software development world can be implemented on hardware in an analogous manner.

The importance of the range and airspace cannot be overlooked in the effective testing of autonomy. The range must be able to supply the necessary airspace to adequately allow the autonomous task to be executed. Additionally, the range must be able to accommodate the test infrastructure, radio frequency spectrum environment, and all of the other logistics required to execute the autonomous test and collect the necessary data.

<u>Surrogate Test Platform's Role in Test Efficiency.</u>

A major driver in test schedule slip is the limited availability of ranges and aircraft. Oftentimes, because of the small windows of availability, large test events are squeezed into specific times with little flexibility. This rigid paradigm common for flight test is incompatible with the rapid test requirements needed to fully test autonomy. The use of appropriately sized aircraft commensurate with the maturity and objectives of the autonomy being tested is the solution to this problem. Smaller UAS are cheaper to operate, more readily available, and require less airspace. Therefore, if certain data requirements can be met on these smaller aircraft, it allows for greater test flexibility. This added flexibility increases the amount of testing that can be performed, which preserves the test schedule without compromising on test effectiveness or safety. Furthermore, maximizing the use of smaller UAS to accomplish objectives can minimize the costs associated for both the test platform and the range. Eventually, a test of autonomy on its intended full-scale platform will need to occur; however, the scope of that test can be significantly reduced because of all the sub-scale testing that has already occurred.

PLANNING & ANALYSIS

General Dwight D. Eisenhower famously said, "… plans are useless, but planning is indispensable." This insight is profoundly applicable to testing autonomy. To be compatible with a responsive, agile development cycle, test planning should be done in such a way that favors multiple, small duration test events. These smaller, more frequent, test events enable testers to experience the performance of the autonomy quicker and provide needed data and feedback to developers sooner. Autonomy testing often involves many "unknown unknowns" and therefore the quickest way to figure out what to do is to try. The ability to successfully execute this type of test strategy hinges on the four previously discussed key components of RTA, LVC, OSA, and surrogate test platforms. The test infrastructure and approach to test execution will obviously shape the planning of the autonomy testing. Regardless, the principle of flexible test planning that emphasizes smaller, more focused test events remains. To maximize the success of test, an autonomy test plan must simultaneously provide concrete objectives while allowing for an iterative approach to "what" and "how" that looks like. To facilitate this iterative approach, the analysis and reporting procedures must likewise be flexible and responsive. Therefore, the traditional linear approach to planning, executing, and then analyzing must be reconfigured to a spiral approach where the analysis of a previous phase feeds into the planning for the next.

Some specific considerations are presented in this section to help provide greater clarity of what a successful autonomy test plan and analysis process looks like. First, before even starting to plan the actual test, the plan for data management must be considered. Next, the concept of Design of Experiments (DOE) is discussed as a method for systematically figuring out what to test and what parameters matter. Following that, incorporating operational testing perspectives into the planning process is discussed. Included in these considerations is discussion on how to incorporate human factors early on and how to best leverage HIL simulations. Next, the importance of factoring security considerations into the planning process is covered. Lastly, a new approach to safety planning called systems theoretic process analysis (STPA) is presented.

Data Management:

While data collection is central to any test initiative, for testing autonomy it is especially important. Data from both simulation and flight test is crucial to training and understanding the behavior of the autonomy in performing a task or a mission. Particularly for autonomy test, detailed logs should be kept for each service for "white box" analysis to help identify why the autonomy took a particular action while performing a task or a mission. The large amounts of data resulting from detailed logs that are collected during test produces some challenges with data management and storage. If the infrastructure is not in place to handle this data production, then the ability to test and evaluate the autonomy will be severely hampered. Some questions that should be asked during the planning stage are:

- How will the data be collected and stored?
- Who will manage the data?
- What security measures, classification issues might arise?
- What are the best data formats for analysis and storage?
- How will flight test data be relative to simulation data?

Understanding and planning for test data management is an essential precondition for a successful test of autonomy. Doing it right initially allows for a more seamless implementation of a spiral approach of planning, executing, and analyzing. If the data management piece is not considered, there will be significant downstream implications, including not being able to accurately test and evaluate the system.

Design of Experiments (DOE):

DOE is a powerful tool to optimize test and evaluation efforts. Used properly, DOE can reduce the number of required test points (and thus test events) while ensuring enough data is collected to identify and characterize statistically significant effects. DOE is especially salient for autonomy testing because of the complexity of the system and the unknown unknowns. Identifying which parameters are valuable from a test and evaluation perspective is oftentimes non-intuitive. Using DOE helps bring clarity to what testers need to consider and what is truly important. As the planning, execution, and analysis cycles progress, greater fidelity models and more specific questions can be asked.

Properly implemented DOE consists of an iterative process where initial designs and tests are used to determine what factors and levels are likely important. These initial designs have a large design space with many factors and levels, and produce low fidelity results. These results cannot be used to characterize system performance without further testing; however, they are critical in reducing the design space and increasing the fidelity of subsequent tests. Initial test designs are best suited for the earliest test events and help to answer the question, "What should be tested?"

Following initial tests, intermediate test designs are generated and executed in an effort to further refine the design space and maximize the impact of each test point. Intermediate designs can build upon the data gathered during initial tests, explore an entirely new area, or (more commonly) a mixture of both. Intermediate testing and design should be iterated and further refined as often as necessary to answer, "What factors and levels have an effect on the system?"

Final test designs are informed by data from initial and intermediate tests. Only factors and levels that have a significant impact on system performance are considered, and the design space is constricted to only encompass the envelope where those factors and levels have a significant effect. The final tests continue to build upon initial and intermediate test data, filling in the blanks where necessary, and re-testing certain test points to verify that previously collected data is still relevant. This final testing answers the question, "What is the impact on system performance of these relevant factors and levels?"

Operational Perspective:

Traditional operational test plans are broken down into Measures of Effectiveness (MOE) and Measures of Suitability (MOS). The added dynamic of autonomous tasks may require MOE/MOSs that would be analogous to pilot evaluations, where performance measures (generally subjective in nature) are rated against minimum standards, and repeated until the minimum standards are met. Furthermore, test teams may require ratings from a number of evaluators in order to come up with a more objective measure. One way to do this is to gather instructors from various units to rate the system under test. As an example, autonomous task measures may include the following graded elements, adapted from an operational unit's Mission Qualification Training (MQT) Syllabus, shown in table 1 with criteria given in table 2.

Table 1 Example Graded Elements for Evaluating Autonomy[4]

Graded Elements	Event 1	Event 2	Event 3	Event…
1) Safety				
2) Crew Coordination				
3) Communication				
4) Risk Management				
5) Decision Making				
6) Navigation				
7) Battlespace SA				
8) Terrain/Wx/Airspace Awareness				
9) Mission Specific TTPs (i.e., ISR, CAS, etc.)				
10) Weapons Employment				
11) Coordinated Attack				
12) Emergency Management				

Note: adapted from pilot MQT syllabus.

Table 2 Example Grading Criteria for Graded Elements

Grade	Explanation of Grade
U	Unknown; performance was not observed; element was not performed
D	Dangerous; performance was unsafe. Any element graded ‘D’ results in overall failure of test, and immediate termination of the test event.
0	Performance demonstrated significant lack of ability
1	Performed task with limited ability; required significant assistance from evaluator.
2	Performed task, but not to the desired levels of speed, accuracy, or safety; required some assistance from the evaluator. Performed most tasks without assistance.
3	Performed task at the desired level of speed, accuracy and safety; required no assistance from the evaluator.
4	Performed task with high degree of ability; exceeded desired levels of speed, accuracy and safety.

Note: adapted from pilot MQT syllabus.

These measures were designed for human pilots and assumes the ability for evaluators to provide feedback on performance between events. They are by no means all-inclusive, but this may provide a framework for measuring inherently subjective measures as objectively as possible. Due to the relatively unpredictable nature of autonomy, test teams may need to account for failure of these measures, and plan for rapid design iterations for re-testing. The importance of establishing strong working relationships with autonomy developers cannot be overstated.

The DT and OT organizations must work together to establish these measures. At early phases of autonomy testing, DT may focus primarily on safe operation (e.g., Safety, Risk Management, Emergency Management, General SA, etc.). Failure of these “safety of flight” measures would be analogous to a “clean

[4] Abbreviations, acronyms, and symbols in figures and tables are defined in appendix D.

kill" in a human pilot checkride. Objective thresholds may be inadequate in measuring these critical areas. An evaluator may exercise judgment and determine that the autonomy acted in a manner that may have potential to cause harm in the future (e.g., too slow to make a decision, flying too close for comfort towards an airspace boundary, acting unpredictably around manned assets, etc.). The test team must be prepared for failures in these areas and ready to enable multiple cycles of updates and retesting until minimum standards, and comfort levels, are achieved.

Furthermore, developing MOE/MOSs will require a level of understanding of how autonomy will be employed on an airborne system. Test teams must engage early with end users and enable continuous feedback with them as they develop or modify tactics, techniques, and procedures (TTPs) for the autonomy under test. These TTPs can then be used to influence the overall test plan. The level of formality (or informality) of this process will be at the test team's discretion, but they must realize that the quality of interaction between users and developers will directly impact the quality of the autonomy under test, not to mention the relevance of the test itself.

Human-System Integration:

How the human will interact with the autonomy is pivotal (Assumption 3) and must be planned for early on in testing. Since the autonomous task is often replacing a specific human function, the focus is shifted towards the technical competence in that task and not the eventual interaction required with the human. While the performance in executing the task is important, the interplay between the human and the autonomy is equally important. This interaction between the human and the autonomy is a topic of widespread research within the human factors community detailed in *Humans and Automation: System Design and Research Issues* (reference 32) and *Human Factors Methods: A Practical Guide for Engineering and Design* (reference 33).

Early on, the human and autonomy user goals must be defined and understood. Assessing and improving these interactions is iterative and often relies on trial and error to optimize the interface. If the human-system interaction becomes an afterthought, which is often the case, the autonomy is susceptible to disuse or misuse. Some of the human factors considerations that should be considered early on are:

- **Operational Environments:** the operating environments of the autonomy and the human interacting with the system. Some considerations that should be made are:
  - Equipment limits (e.g., communication bandwidth and range)
  - Adversarial effects (e.g., degraded communications, GPS jamming)
  - Environmental effects (e.g., temperature effects [both for human and machine], moisture, pressures)
- **Human-Autonomy Interface:** the method that the human will interact with the autonomy must be defined. Things to consider for the interface are:
  - Inputs to the Human:
    - Visual cues (e.g., light in cockpit, display on screen, visual cue from UAS [e.g., wing rock])
    - Tactile cues (e.g., vibration, pressure, texture, etc.)
    - Auditory cues

    - Outputs from the Human:
        - Physical manipulations (e.g., button pushes, knob turns)
        - Biophysical measurements (e.g., heart rate, eye tracking, oxygenation)
        - Verbal Commands

- **Usability:** the ease of interaction with the autonomy, as described in *Usability Assessment: How to Measure the Usability of Products, Services, and Systems* (reference 34). Some factors that impact usability are:
    - Achievability (Can the task be accomplished?)
    - Efficiency
    - Effectivity
    - Easily learnable
    - Generally satisfactory
- **Workload:** how hard the human has to work, mentally and physically, when interacting with the autonomy as described in *Workload Assessment: How to Diagnose Workload Issues and Enhance Performance* (reference 35). Workload is often measured by considering factors like:
    - Subjective Metrics (e.g., temporal demand, spare capacity)
    - Objective Metrics (e.g., biophysical measurements)
    - Performance-based (e.g., task success)
- **Human-Autonomy Tasking:** how shared human-machine tasks are distributed and accomplished. Some factors are:
    - Task Allocation: who has responsibility for accomplishing the task (similar concept to Crew Resource Management in the aviation world)
    - Time on Task: how long it takes to complete the task. This could also measure how long after the task has been completed that it is still being monitored or observed.
    - Task Difficulty: who can do what and when.
    - Task Involvement: defined by attention and arousal required to do a task. Humans do not want either extreme (Yerkes-Dodson law)
- **Situational Awareness:** the awareness of agents (human and machine) within the system including the agents which they interact with (both friend and foe). Situational awareness looks at, among other things:
    - Past, present and future events
    - Local and global awareness
- **Military Utility:** how well the autonomy performs its intended mission. The military utility of an autonomous task must include the human interaction as part of its calculation.

- **Human Bias and Individual Differences:** not every human will respond in the same manner to the autonomy under test. Several things to consider for this factor are:
    - Demographics
    - Predisposition
    - Experience
    - Training
    - Cognitive Ability
    - Physical and Emotional States
- **Trust:** negative outcomes could result from either over- or under-trust. Many models and scales have been developed to measure and quantify trust.

Many of these factors are interdependent and are not easily isolated. This can lead to challenges in explicitly testing certain factors. The four components of RTA, LVC, OSA, and surrogate test platforms all provide different capabilities that can assist in human factors testing.

Of all the factors, trust might be the most salient in predicting the ultimate success of the autonomous task. Autonomy cannot be evaluated in isolation and is fundamentally a joint human-machine enterprise (Assumption 3). If the human does not trust the autonomy, then it will not be able to fully achieve its potential (reference 4). Evaluating the human's role in the implementation of autonomy is essential and must be considered early in the development process. Properly leveraging LVC creates a representative environment that can facilitate valuable feedback on a human's trust towards the autonomy under test. This interaction can be as complicated as a human flying a virtual aircraft with a live autonomous wingman or as simple as a human observing a constructive simulation. The variety of ways that LVC can be applied greatly bolsters the test effectiveness because it gives entry points for T&E throughout the development process. LVC is an invaluable asset in molding and shaping the autonomy's interaction with the human, which will ultimately decide its overall effectiveness.

Role of Hardware-in-the-Loop (HIL):

The use of HIL simulation is a common practice throughout the flight test community. The ability to test software on representative hardware, before flight test, is a natural first-step to mitigate risk. In most contexts, the HIL simulations are performed on subcomponent-level systems, often with safety-critical functions. While this application is still pertinent when considering how the autonomy engine will interact with the other system components, the real power of HIL is manifested when used for mission and task-level planning and analysis. When testing autonomy, the hardware is the autonomy engine, itself. Therefore, leveraging a M&S environment that can accurately stimulate the autonomy engine provides valuable insight into how the system will perform in flight test. Specifically, HIL impacts the autonomy test planning by determining the test readiness and informing the test team of the anticipated behaviors of the autonomy.

Subsystem integration aims to be one of the greatest barriers to testing of autonomy. Assumption 2 proposes that a monolithic "autonomy" is not practical and therefore, future autonomy applications will be comprised of numerous services, possibly designed by different developers, relying on a diverse set of sensors and inputs. Ensuring that each subcomponent is sending and receiving the correct information is pivotal to a successful test. HIL provides an avenue to test and debug problems on the ground accurately and quickly.

Performing a HIL event prior to flight test provides the test team with a "dress rehearsal" and displays what the expected performance of the autonomy will look like. This is especially pertinent, because, unlike a traditional test executed by a human pilot, an autonomy test will feature a higher degree of uncertainty. First, the "right" action of the autonomy is not necessarily intuitive. The autonomy could make a decision similar to a human operator, or could choose something different based on how it's coded. Emergent behaviors are inherent to complex systems. The benefit of HIL is that these emergent behaviors can be first encountered in a simulated environment and provide the test team with insight into how the real-world autonomy will function.

The HIL also features prominently in the analysis of autonomy. The probabilistic nature of the analysis required for T&E of autonomy and the limitations on flight test make HIL a necessary analysis tool to augment live test. Once the HIL simulations are shown to be accurate enough to the real world, advances in supercomputing and data science can be leveraged to provide large amounts of test events. This added data is invaluable for evaluating the system performance and ultimately for its verification and validation.

Security:

Security is especially a concern when it comes to employing autonomy onboard an air platform, primarily because of the decision making capabilities that must reside on the system. The TTPs for any design reference mission (DRM) must be codified in order to design the autonomous task or mission. When these TTPs are coupled with a DRM and a system, this will undoubtedly invoke a higher security classification and could pose problems for the test and evaluation of that autonomous task. Unlike manned aircraft, where the human maintains and controls classified TTP information, the autonomy engine may be more susceptible to exploitation. To counter these risks, foresight is essential to define and identify what is classified early on in the process. This includes thinking through how the various autonomous tasks fits into the overall DRM, how the autonomy engine facilitates internal communication between the various tasks, how the autonomy interfaces with contributing systems, how the test data will be collected and stored, and how M&S assets that contribute to the LVC environment affect security classifications.

Like safety, security holds a high priority. The best way to maintain a secure, yet rapid test approach is to minimize the number of items that require a higher security classification. The four components of RTA, LVC, OSA, and surrogate platforms help mitigate some of the associated security concerns with testing autonomy. In general, if proper planning is done on the front-end of a test program, then the test team can avoid unnecessary delays in testing due to classification and infrastructure limitations.

Security considerations need to be also be included in the analysis phase. Many autonomy test plans may focus primarily on data collection for further autonomy development. In this case, protecting the raw data that will be used for future training and development is paramount. Additionally, the performance and vulnerabilities of the autonomy will be determined in the analysis phase. It is important that this information is protected appropriately to prevent exploitation.

Systems Theoretic Process Analysis (STPA):

The current flight test safety approach relies on system safety analysis, a careful study of historical testing, test safety plans, unexpected test events, and lessons learned from similar test programs. This philosophy looks at each component of a system, identifies the vulnerabilities and builds a risk profile based on these calculations. While this approach has its benefits, some notable downsides are the inability to recognize emergent properties between system interactions, the need for in-depth modelling and engineering analysis, and its difficulty in assessing human-machine interactions.

A new philosophy in safety planning and mitigation called STPA has begun to gain traction in the flight test world. The STPA was initially designed for software, and is particularly promising in the world of autonomy and complex systems. The general premise of STPA is that it leverages a "top-down," systems engineering approach to identify hazards in complex systems as described in *System Theoretic Process*

Analysis Handbook (reference 36) and "A Comprehensive Safety Engineering Approach for Software-Intensive Systems based on STPA" (reference 37). The general method for using STPA is shown in figure 4 and is comprised of four steps: 1) Define Purpose of the Analysis, 2) Model the Control Structure, 3) Identify Unsafe Control Actions, and 4) Identify Loss Scenarios.

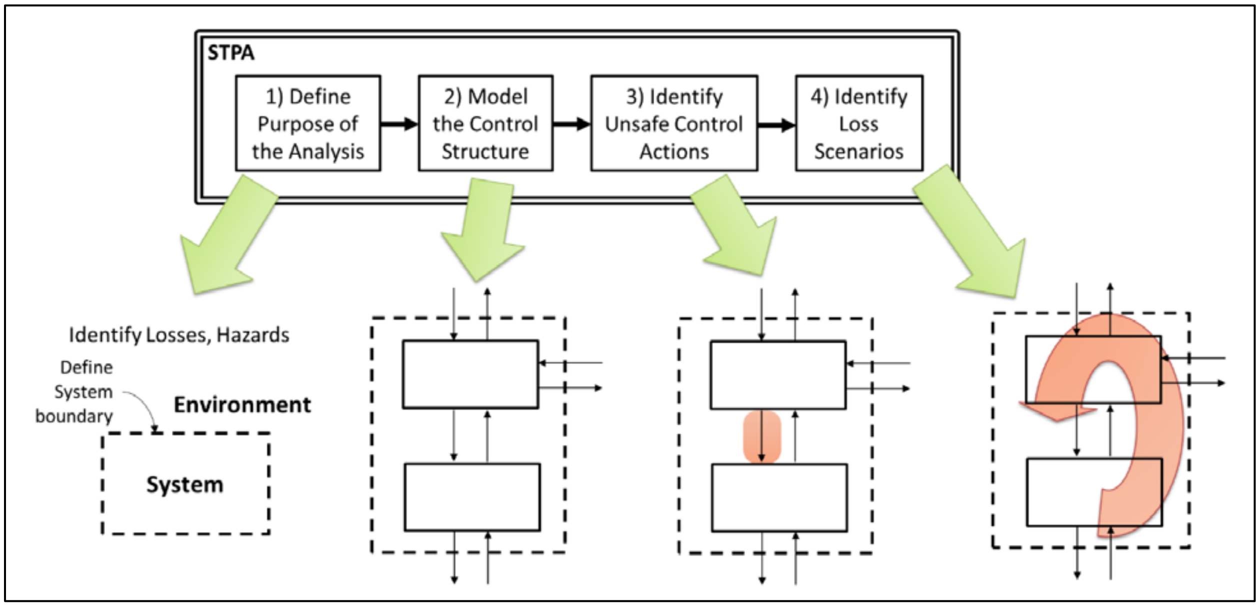
STPA
1) Define Purpose of the Analysis
2) Model the Control Structure
3) Identify Unsafe Control Actions
4) Identify Loss Scenarios
Identify Losses, Hazards
Define System boundary
Environment
System

Figure 4 Overview of General STPA Process (reference 36)

In step 1, the losses and hazards are identified. This can pertain specifically to safety of flight hazards, but also generalizes to other fields like performance, security, organizational communication, privacy, etc. Step 2 looks at the different components within the system and identifies what the various inputs and outputs are to each component. In step 3, unsafe control actions (UCAs) are developed to address how the control actions from step 2 could result in one of the identified hazards listed in step 1. These UCAs help derive functional requirements and constraints for the system. Finally, step 4 builds scenarios that help explain how these UCAs might occur in the system. Specifically, these scenarios identify what breakdowns in the flow of information (incorrect feedback, inadequate requirements, component failures, etc.) could lead to an UCA. Additionally, the scenarios look at areas where a correct control action could be provided but not properly implemented.

Given the systems engineering nature of STPA, this method excels for complex, interconnected systems. Additionally, STPA is perfect for deriving system requirements and provides a holistic view of how each part contributes to the overall system and where potential failure states lie. This attribute allows for easy consideration of human-machine interactions, which is a pivotal component in autonomy testing.

The STPA is most effectively applied during the requirements development and early design of new systems, as the scenarios and unsafe control actions identified may readily be mitigated by the specification of new requirements early enough to effect design changes to the system. In flight test, there is rarely capacity or appetite for implementing new requirements without adequate justification. Generally this justification is strongest when there is the actual evidence of the unsafe scenario; flight testers are rarely successful in driving new requirements prior to the occurrence of unusual test events. Furthermore, as STPA is not a probabilistic systems analysis, it is difficult to ascertain the likelihood of unsafe control actions. In traditional risk assessment and mitigation, the probability of the hazard occurring is half of the overall risk calculation (the other half being the consequence). The lack of a probability of occurrence/failure makes it hard to accurately discern which of the hazards identified by STPA should be preferentially mitigated and which ones can be ignored by virtue of their low probability. On top of that, the process of implementing STPA is fairly rigorous and can easily be misapplied by novice users. To avoid this pitfall, a cross-functional team must go through the process under the direction of a skilled facilitator.

STPA may be useful for the test and evaluation of new systems that perform autonomous tasks and missions. These test programs may benefit the most due to the complex nature of the system, the novel interaction with human operators, and the lack of historical test programs from which to draw lessons learned. Additionally, STPA can more broadly be applied to evaluate the factors that might contribute to the unsuccessful completion of the autonomous task. Instead of solely focusing on safety of flight issues, the STPA process could evaluate the linkages between the test planning process, the integration of different autonomy services, and the human interactions to identify scenarios that might result in an unsuccessful test.

There is no "one size fits all" approach that can be magically applied to safety planning for autonomous and complex systems. STPA is presented in this handbook because of its unique ability to evaluate interactions between and among complex systems. There is still a definite need for the traditional risk assessment tools, but these should be considered in light of the larger, system and subsystem level interactions.

PARADIGM SHIFT: AGILE DEVOPS

The 2018 National Defense Strategy (reference 1) calls the DoD to "build a more lethal force" and to "reform the DoD for greater performance and affordability." To truly implement these directives for autonomy, the test community needs to integrate more fully with the autonomy development community (both military and contractor) and the end users who will operate these systems. Test plays a central role and acts as the bridge between the development of the new autonomous tasks and the fielding and operation on airborne systems. The new paradigm that is essential for advancing the state of autonomy T&E to the next level is one of continuous learning, testing, and fielding.

In this new paradigm, the traditional, waterfall development model as an acquisition strategy becomes intractable and instead is replaced with an Agile strategy. Figure 5 shows how the waterfall and Agile approaches differ, with the emphasis on how the Agile process best manages risk in implementation of autonomous tasks and missions.

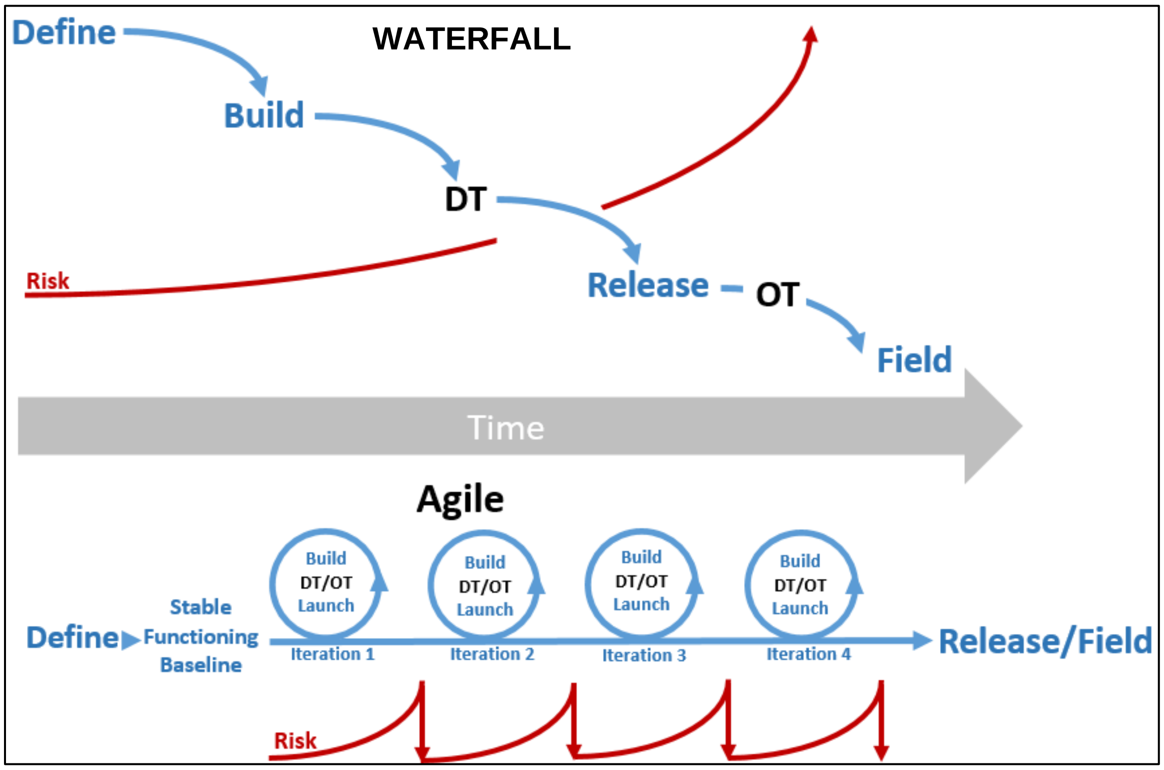

Figure 5 Waterfall vs. Agile Software Development Risk Profiles as described in *Air Force Test and Evaluation Guide* (reference 38)

The key to unlocking this new paradigm is the concept that "autonomy" is fundamentally software (Assumption 1). Once this assumption is internalized, the autonomy itself can be properly decoupled from a specific platform. This focus on software development, allows the test community to learn from and implement analogous best practices seen in industry. Agile and DevOps processes are two specific practices that could dramatically boost the test and evaluation capabilities for autonomy. In the proceeding subsections, high-level descriptions of Agile and DevOps are presented to give some context for how they could impact the T&E of autonomy. Implementing the principles of Agile and DevOps will be difficult, so the final section outlines some barriers that need to be overcome to best respond to the challenges to testing autonomy.

Agile:

The agile software practice provides a rapid and effective means of generating software that addresses the warfighter's needs. While the test community will not be developing autonomy software, it must be compatible with this process in order to provide an effective T&E capability. Tremendous advancements in modern software development have been made through the agile software development philosophy. The best way to define the Agile process is by citing their Manifesto (reference 22):

- Individuals and Interactions OVER Processes and Tools
- Working Products OVER Comprehensive Documentation
- Customer Collaboration OVER Contract Negotiation
- Responding to Change OVER Following a Plan

While the emphasis of Agile runs counter to the traditional test philosophy, these principles must be embraced to effectively implement autonomy onboard a platform. It is important to note that Agile prioritizes the items on the left over the items on the right. This does not mean that processes and tools, comprehensive documentation, contract negotiations, and following a plan do not have their role. It does, however, mean that these items should not come at the expense of the items on the left.

Adapting the three phases of test to test autonomy is a good starting point for showing how the Agile principles can begin to work within the context of an AFTC test framework. Figure 6 shows a diagram of how the test community can operate within an Agile software development environment. In this figure, a prioritized backlog is maintained that is based on stakeholder requirements, CONOPS, threats, and stable software baselines. Coupled with test strategy and resources, the backlog feeds an iterative software development cycle. The test team and users operate within this cycle to produce an intended capability. The performance of this capability serves as the launching point for the next cycle. The process continues indefinitely as new capabilities are continually delivered and improved.

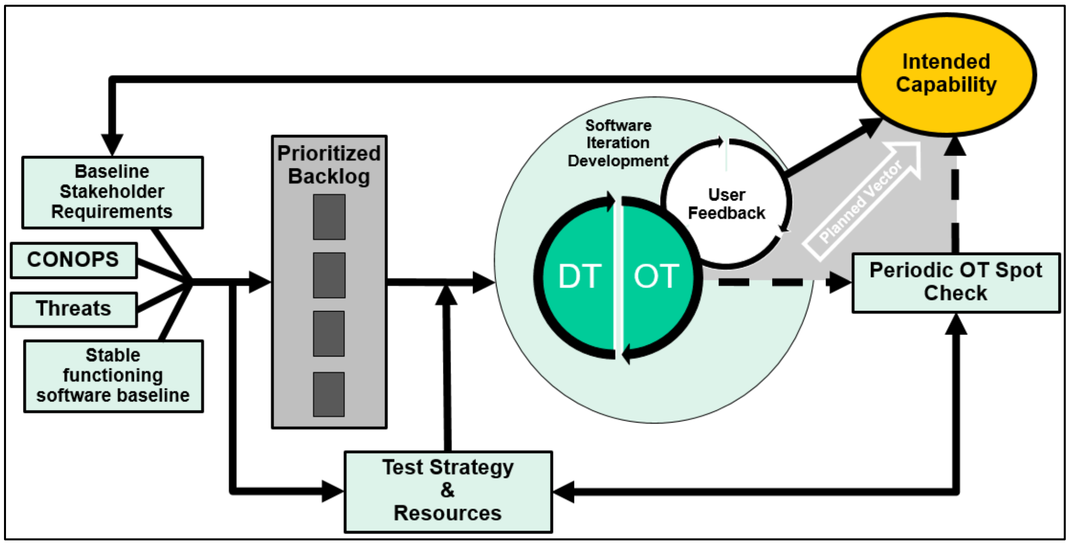
Intended Capability
Baseline Stakeholder Requirements
CONOPS
Threats
Stable functioning software baseline
Prioritized Backlog
Software Iteration Development
User Feedback
DT
OT
Planned Vector
Periodic OT Spot Check
Test Strategy & Resources

Figure 6 Test Relationship in ASD [38]

DevOps:

DevOps is defined as "an organizational shift in which, instead of distributed siloed groups performing functions separately, cross-functional teams work on continuous operational feature deliveries" in

“DevOps,” from IEEE Software (reference 39). In any organization, the development and operations functions have competing interests. Developers in the Air Force (i.e., program offices, contractors, and research organizations) constantly seek to provide new functionality that will benefit the customer (i.e., best complete the mission). On the other hand, the operations side seeks stability and reliability. The challenge for the test community is to facilitate the rapid and innovative development of new and updated autonomous tasks, while ensuring the operational community has reliable and trustworthy systems. DevOps was designed to leverage new technological advances in computing and automation to symbiotically merge these two competing interests.

To better understand the DevOps concept, it is better to decompose DevOps into some of the key practices that help define it. Five key practices of DevOps as defined in “What is DevOps” (reference 40) are:

- Continuous Integration: Instead of waiting for a new autonomy service or task to be completely finished before merging with the master branch, continuous integration uses a shared repository that is continually updated. This allows for better collaboration with other developers and also helps identify potential problems sooner. Unit testing and automation are leveraged to ensure stable code and find potential bugs. The result is that the autonomy task or service is produced faster without sacrificing quality.
- Continuous Delivery: Updates to the autonomy are automatically pushed to a test environment once they are ready. This test environment could range from a purely simulated (virtual and constructive) environment to a complex LVC test using numerous live assets.
- Microservices: This builds on the service-oriented architecture approach and allows for the fragmentation of complex applications. The use of microservices within an OSA allows for a modular approach to both design and implementation of autonomous tasks.
- Monitoring and Logging: During both the testing and fielding of autonomy, the performance of the system is monitored and issues are logged. This feedback helps improve the next iteration of updates and developments and contributes to trend analysis.
- Cross Functional Collaboration: The whole DevOps construct does not work if there is not good communication and collaboration amongst the various stakeholders. Even within the test community, it is essential that experts in computer science, human factors, engineering (aeronautical, electrical, computer, systems, etc.) and logistics be represented and effectively work together.

The optimal application of DevOps is through the use of a cloud infrastructure. A cloud-based application is crucial for timely, secure, and effective management of software as it progresses along its development. Additionally, the success of DevOps is contingent on the successful use of automation for building, managing, testing, and analyzing new software. Functions that are repeatable, predictable, and time-intensive should be automated. This helps reallocate the human brain power to solving the more complicated and abstract problems that arise with testing autonomy.

Barriers to Overcome:

Fully adopting an Agile DevOps approach within the AF will be difficult to implement. However, the purpose of this section is to provide a vision of the optimal environment for testing and evaluating autonomy. In striving towards implementing a DevOps approach, there are several barriers that first must be overcome.

The first is cultural barriers is the mindset within the test community. There is a temptation to rely on the “business as usual” approach when it comes to testing autonomy. The problem with this approach is that the level of complexity involved in testing autonomy is not practically feasible using the old approaches. Additionally, the use of more technology and automation might alter the makeup and roles of the traditional test team. It is important to remember that the human plays a pivotal role in both the implementation and testing of autonomy. Just because a task once performed by a human (e.g., data analysis, mission planning) becomes automated, does not mean that the human is no longer needed. Human capital is a precious resource and these new concepts of Agile and DevOps are intended to first, and foremost, maximize human potential.

More generally, the stove-piped approach of separate DT and OT, as well as involving the user at the end of the acquisition is not sufficient. Testers must be more in touch with the user communities. A future approach might feature embedding testers with operational units to help streamline the spiral development and testing of new capabilities.

The second cultural barrier is the investment priority of the Air Force, in general, and the test community, in particular. Traditionally, the Air Force has had a "hardware-first" mindset and often has paid for state-of-the-art military hardware at the expense of its communications and information technology (IT) infrastructure. Facilitating an environment conducive to safely, securely, effectively, and efficiently testing and evaluating autonomy means that the financial investment must shift to prioritizing software and its corresponding infrastructure. This means a renewed prioritization is required for cloud computing resources, data management tools and logistics, and investing in personnel who are experienced in data science, computer science, and IT.

The last barrier is the problem of vendor lock and the current acquisition paradigm. The status quo for most military systems is that a single contractor provides an end-to-end solution (both hardware and software) that provides a set of capabilities and the requirements for these capabilities must be determined multiple years ahead of time. It is difficult to accurately forecast what specific capabilities are needed multiple years in the future. Inevitably, changes to the military system must be made to stay operationally relevant. These changes are both time intensive and expensive because all of the system software and its interfaces tend to be proprietary information of the prime contractor. This model is antithetical to meeting the objectives of the National Defense Strategy.

To help promote a test approach conducive to testing autonomy, the test community must advocate for government-owned interfaces that avoid "vendor lock". To accomplish this task, the larger acquisition community needs to create a new value proposition for military contractors that makes it worthwhile for them to decouple the hardware and software development. A great example from the commercial sector is the Apple application (app) store. Apple provides the hardware (the iPhone) and a software development kit (SDK) for building an app compatible with their operating system. The SDK allows third party developers to write software that seamlessly runs on the iPhone. Through this model, the hardware developer (Apple), software developers, and end users all share in a symbiotic relationship that provides a timely and capable system. Likewise, for autonomy development, a similar construct would promote a mutually beneficial relationship for the military, traditional military contractors, and entrepreneurial software developers.

The bottom line is that change is hard. One of the purposes of this handbook is to equip testers with tools to adapt the solid foundation of current test practices to best accommodate the testing of autonomy. The principles of the Agile and DevOps philosophies provide the ideal framework for quickly producing high-quality autonomy software.

CONCLUSIONS

As the use of autonomy on air platforms become the new normal of warfare, it is essential that appropriate test and evaluation methods are embraced to maximize safety, security, effectiveness, and efficiency. Testing and evaluating autonomy is a complex problem that requires a holistic approach from a cross-functional team. This handbook outlined best practices, insights, and tools for the testing and evaluating of autonomy. Generally, the approach for autonomy test should feature early user involvement, continuous and cumulative feedback, streamlined processes and products, a pilot training approach, and consideration of human-machine interaction.

The testing of autonomy was first examined using the current paradigm of the three phases of test (planning, execution, and analysis). The execution phase of an autonomy test was first discussed because of its various challenges. Leveraging Run Time Assurance, Live-Virtual-Constructive capabilities, an Open Systems Architecture, and surrogate platforms presents the best way of achieving a test that is safe, secure, effective, and efficient. These four components all individually contribute to the safety, security, effectiveness, and efficiency and collectively form a formidable framework for executing autonomy tests. The requirements to successfully execute an autonomy test informed the planning and analysis processes. The biggest takeaway for both of these phases is that an iterative approach is essential. The wide array of “unknown unknowns” makes it nearly impossible to have a one-time, linear approach to test planning and analysis. Instead, tools like design of experiments, hardware in the loop, operational and human factors considerations, and systems theoretic process analysis all provide an avenue for iteratively and successfully completing both phases.

The philosophies of Agile and DevOps were presented to optimally test and evaluate autonomy. The concept of DevOps integrates the development and operations of autonomous capabilities using a cross-functional team. In this paradigm, the stove-piped acquisition, test, and operations process is replaced by a process with continuous integration, continuous delivery, microservices, continuous monitoring and logging, and cross-functional collaboration. Additionally, a test capability that is compatible with the agile software development process must be embraced. The key principles of an Agile test capability are favoring interactions and individuals, working products, customer collaboration, and responding to change. Lastly, several cultural barriers that will inevitably arise when implementing some of the Agile DevOps principles were discussed. The bottom line is that if the objectives of the 2018 National Defense Strategy are to be realized then hard and dramatic changes are inevitable.

Much like the iterative nature required to properly test and evaluate autonomy, this handbook is the first iteration. As more autonomy programs emerge and autonomy tests are conducted, those lessons learned and tools will be gathered and presented in the next version.

REFERENCES

1. "Summary of the National Defense Strategy of the United States of America," Office of the Secretary of Defense, Washington D.C., 2018.
2. John S. McCain National Defense Authorization Act for Fiscal Year 2019, H.R. 5515, 115th Congress, 2018.
3. "Autonomous Horizons: The Way Forward," Office of the United States Air Chief Scientist, *Air University Press*, Maxwell AFB, Alabama, 2019.
4. "Autonomy Community of Interest (COI) Test and Evaluation", *Test and Evaluation Verification and Validation (TEVV) Working Group: Technology Investment Strategy 2015-2018*, Office of the Assistant Secretary of Defense (Research and Engineering), Washington D.C., 2015.
5. Bradshaw, J., R. Hoffman, M. Johnson and D. Woods, "The Seven Deadly Myths of 'Autonomous Systems,'" *IEEE Intelligent Systems*, vol. 28, no. 3, pp. 54-61, 2013.
6. Bar-Yam, Y., "Dynamics of Complex Systems", Reading: Addison-Wesley, Routledge, New York, 1997.
7. Bell, M. "Service-oriented Modelling", John Wiley & Sons, Inc., New Jersey, 2008.
8. Davis, J., R. Mayer, and D. Schoorman, "An Integrative Model of Organizational Trust," *Academy of Management Review*, vol. 20, no. 3, pp. 709-734, New York, 1995.
9. Kaminski, P., R. Murphy, and J. Shields, "The Role of Autonomy in DoD Systems", Defense Science Board, Washington D.C., 2012.
10. David R., C. Fields, and P. Nielson, "Defense Science Board: Summer Study on Autonomy", Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics, Washington, D.C., 2016.
11. Boyd, J., "A Discourse on Winning and Losing", Maxwell AFB, AL: Air University Library Document M-U 43947 (Briefing Slides), 1987.
12. "Assessment of Test and Evaluation Infrastructure and Improvements Required for Future Emerging Technologies", Office of the Under Secretary of Defense for Research and Engineering, Department of Defense, Washington, D.C., 2019.
13. Clark, M., X. Koutsoukos, R. Kumar, I. Lee, G. Pappas, L. Pike, J. Porter and O. Sokolsky, "A Study on Run Time Assurance for Complex Cyber Physical Systems", Air Force Research Laboratory, Wright-Patterson AFB, Ohio, 2013.
14. Clark, M., J. Cooper, M. DeVore, N. Gandhi, K. Horneman, , N. Richards, J. Schierman, Smolka, S, and S. Stoller, "Run Time Assurance for Complex Autonomy," *AFRL Case No. 88ABW-2015-2813*, Wright-Patterson AFB, Ohio, 2015.
15. Chowdhury, O., A. Datta, A. Kane, and P. Koopman, "A case study on runtime monitoring of an autonomous research vehicle (ARV) system," *Runtime Verification*, pp. 102-117, Vienna, Austria, 2015.
16. Avram, R., M. Clark, J. A. Muse, and X. Zhang, "Nonlinear Adaptive Control of Quadrotor UAVs with Run-Time Safety Assurance," in *AIAA Guidance, Navigation, and Control Conference*, Grapevine, Texas, 2017.
17. Clark, M., A. Fifarek, K. Gross, J. Hoffman, K. Rattan, E. Swenson, L. Wagner, and M. Whalen "Formally Verified Run Time Assurance Architecture of a 6U CubeSat Attitude Control System," in *AIAA Inotech@Aerospace*, San Diego, California, 2016.

18. Bakun, M., K. Gahan, C. Gotwald, M. Hammond, R. Kolesar, and S. Mak Jian Ming, “A Limited Evaluation of an Automatic Ground Collision Avoidance System for Performance Limited Aircraft,” USAF Test Pilot School, Edwards AFB, California, 2018.

19. LaPlant, N., K. LeBay, N. Olson, J. Rhodes III, and T. Vance, “Operate Remote Aircraft Clairvoyantly in a Limited Evaluation (ORACLE),” USAF Test Pilot School, Edwards AFB, California, 2014.

20. Bearce, J., R. Chrash, F. Meyer, R. Meyer, and E. Sink “Flight Test Evaluation of the HAVE RAIDER II Autonomous Wingman Architecture,” USAF Test Pilot School, Edwards AFB, California, 2017.

21. Allain, C., S. Pontzer, M. Rowan, B. Tillman, and J. Tekell, “Flight Test Evaluation of the AFRL Automated Wingman Functions,” USAF Test Pilot School, Edwards AFB, California, 2015.

22. Beck, K., M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, and R. Jeffries, “Manifesto for Agile Software Development”, Agile Alliance, Coryton, Tennessee, 2001.

23. “Modeling and Simulation (M&S) Glossary,” Department of Defense Modeling and Simulation Coordination Office, Alexandria, Virginia, 2011.

24. Esper, M., R. Spencer, and H. Wilson “Modular Open Systems Approaches for our Weapon Systems is a Warfighting Imperative,” in Memorandum for Service Acquisition Executives and Program Executive Officers, Washington D.C., 2019.

25. “Modular Open Systems Approach,” Office of the Deputy Assistant Secretary of Defense for Systems Engineering, Washington D.C., 2017.

26. Humphrey, L., D. Kingston and S. Rasmussen, “A Brief Introduction to Unmanned Systems Autonomy Services (UxAS),” in 2018 International Conference on Unmanned Aircraft Systems, Dallas, Texas, 2018.

27. Kingston D., S. Rasmussen, “afrl-rq/OpenUxAS,” Github Repository, accessed 30 Apr 2020. URL: https://github.com/afrl-rq/OpenUxAS.

28. Villarreal, J., “U.S. Air Force Research Lab Awards GE TEAMS Program,” *Scientific Systems*, Woburn, Massachusetts, January 2019.

29. Chong, E., C. Eaton, and A. Maciejewski, “Services-Based Testing of Autonomy (SBTA),” *The ITEA Journal of Test and Evaluation,* vol. 38, pp. 40-48, Fairfax, Virginia, 2017.

30. Jenkins M., L. Zieglar, “Commercial National Security Algorithm (CNSA) Suite Profile of Certificate Management over CMS,” 28 September 2018, accessed 24 September 2019, URL: https://tools.ietf.org/id/draft-jenkins-cnsa-cmc-profile-00.html.

    See appendix A for a copy of this reference.

31. “Unmanned Aircraft System Airspace Integration Plan,” Office of the Secretary of Defense, Department of Defense, Washington D.C., 2009.

32. Sheridan, T., “Humans and Automation: System Design and Research Issues”, Santa Monica: John Wiley & Sons, Inc., New Jersey, 2002.

33. Baber, C., D. Jenkins, P. Salmon, N. Stanton, L. Rafferty, and G. Walker, “Human Factors Methods: A Practical Guide for Engineering and Design,” *CRC Press*, Routledge, New York, 2017.

34. Kortum, P., “Usability Assessment: How to Measure the Usability of Products, Services, and Systems,” *The Human Factors and Ergonomics Society*, Santa Monica, California, 2016.

35. Matthews, G., L. Reinerman-Jones, “Workload Assessment: How to Diagnose Workload Issues and Enhance Performance,” *The Human Factors and Ergonomics Society*, Santa Monica, California, 2016.

36. Leveson N., J. Thomas, “System Theoretic Process Analysis Handbook”, MIT, Cambridge, Massachusetts, 2018 (URL: http://psas.scripts.mit.edu/home/materials/).

37. Abdulkhaleq, A., N. Leveson, and S. Wagner “A Comprehensive Safety Engineering Approach for Software-Intensive Systems based on STPA,” *Procedia Engineering*, vol. 128, pp. 2-11, Elsevier, Amsterdam, Netherlands, 2015.

38. “Air Force Test and Evaluation Guide”, United States Air Force Test and Evaluation, Washington D.C., 2019.

39. Ebert, C., G. Gallardo, J. Hernantes and N. Serrano, “DevOps,” *IEEE Software*, vol. 33, no. 3, pp. 94-100, New Hampshire, 2016.

40. “What is DevOps,” Amazon, [Online]. *Amazon Web Services,* accessed 26 September 2019, URL: https://aws.amazon.com/devops/what-is-devops/.

    See appendix A for a copy of this reference.

41. Clive, P., J. Johnson, B. Birkmire D. Hodson, M. Moss, and J. Zeh, “Advanced Framework for Simulation, Integration and Modelling,” *In Proceedings of the International Conference on Scientific Computing (CSC)*, Nevada, 2015.

APPENDIX A – BIBLIOGRAPHY

ADDITIONAL READING

The authors recommend the following as additional reading material. There are many groups tackling the concept of autonomy test, and these resources address some topics not covered within the scope of this handbook.

1. DoD Directive 3000.09, *Autonomy in Weapon Systems.* This directive:
    a. Establishes DoD policy and assigns responsibilities for the development and use of autonomous and semi-autonomous functions in weapon systems, including manned and unmanned platforms. (DoDD 3000.09, 1. a.)
    b. Establishes guidelines designed to minimize the probability and consequences of failures in autonomous and semi-autonomous weapon systems that could lead to unintended engagements. (DoDD 3000.09, 1. b.)
2. DAU CLE 002, *Introduction to the Test and Evaluation of Autonomous Systems*
3. IDA Document NS D-9266, *Operational Testing of Systems with Autonomy,* Wotjon, Heather M., et al., Institute for Defense Analyses, March 2019
4. MP180941, PR-17-2408, *Human-Machine Teaming Systems Engineering Guide,* McDermott, Patricia, et al., MITRE Corporation, December 2018
5. Murphy R., J. Shields, "The Role of Autonomy in DoD Systems," Defense Science Board, Washington D.C., 2012.
6. Huttermann, M., DevOps for Developers, Apress, 2012.
7. Anno, G., L. Brien, E. Fleishman, V. Gawron, E. Jones, E. Lovesey, L. McGlynn, G. McMillan, R. McNally, and D. Meister "Proceedings of the Human Factors Society Annual Meeting," *SAGE Publications*, vol. 35, pp. 1284-1287, 1991.
8. Brewer, M., J. Eggstaff, S. Roberts, P. Sohl, and J. Tyler, Operational Test Agencies Six Core Test Principles, Department of Defense, Washington D.C., 2019.

This list is not all-inclusive, and we recommend the reader explore all available literature.

ATTACHED REFERENCES

(Double click the paperclip icon to open the file.)

30. "Commercial National Security Algorithm (CNSA) Suite Profile of Certificate Management over CMS"

40. "What is DevOps"

This page intentionally left blank.

APPENDIX B – AUTONOMY TEST CAPABILITIES

TESTING OF AUTONOMY IN COMPLEX ENVIRONMENTS (TACE)

System Description:

The Testing of Autonomy in Complex Environments (TACE) system was developed by the Johns Hopkins University Applied Physics Lab (JHU/APL) as a tool for safely testing autonomy onboard a platform. TACE contains an RTA and LVC functionality, it was designed to be compatible with an OSA, and interfaces with the Pixhawk autopilot. A high-level depiction of the TACE software architecture and its interfaces with the autonomy engine and the platform interface is presented in figure B1.

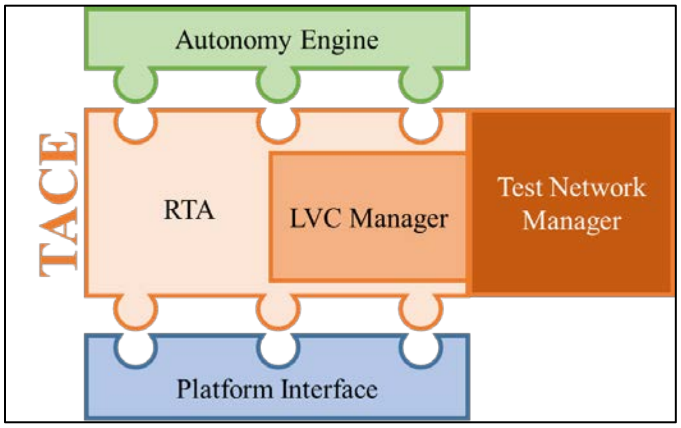

Figure B1 TACE Software Configuration

This figure highlights two important facets of TACE. First, that TACE was designed as test middleware located between the autonomy engine and the platform interface to the autopilot. This is important because it guarantees that nothing commanded from the autonomy can circumvent TACE and command the platform directly. If the autonomy commands the platform to do something that will violate a safety of flight parameter, TACE’s RTA function will unilaterally stop the commands from the autonomy and send a remediation command to the platform instead. As of September 2019, the three safety of flight parameters that are monitored are geospatial boundaries, proximity with other test aircraft, and communications with the TACE ground station. These constraints are tailorable and can be programmed before flight or manipulated in real-time via the TACE ground station. The remediation action for the current version of TACE is to return to a pre-determined remediation waypoint and loiter indefinitely until commanded otherwise by the ground station.

Additionally, the location of TACE in the overall test configuration allows for a natural inclusion of LVC. While TACE monitors commands being sent to the platform it is also responsible for passing aircraft state information back to the autonomy engine. Therefore, information regarding synthetic entities, simulated sensor data, and other effects can be sent by the LVC to the autonomy to stimulate its decision making process without impacting any safety of flight or test parameters of the platform. For the current version of TACE, the Advanced Framework for Simulation, Integration, and Modelling (AFSIM) (reference 41) is used as the LVC synthetic force generator (SFG). Lastly, the test network manager allows the test team direct access to TACE during flight test to monitor the safety of flight parameters, manage the LVC, and log all of the data being sent between the autonomy engine and the platform.

The second important feature of TACE, symbolized by the puzzle piece shapes in figure B1, is that TACE was designed to be modular. The use of an OSA, like UxAS, allows for different types of autonomy engines and platforms to be used with TACE for test. Furthermore, after the autonomy has been verified and approved to fly in an operational system, then TACE can be removed and the autonomy can seamlessly interface with the platform directly.

SUMMARY OF FLIGHT TEST EXPERIENCE

TACE was first flown on 26 February 2019 (example shown in figure B2), and the initial test event featured eight autonomous test sorties (four of which were multi-ship) aimed at demonstrating the TACE system. The three specific test objectives for the TACE flight test were to characterize the RTA performance, demonstrate the LVC functionality, and to demonstrate the TACE ground operator's control of the system. These three test objectives can be traced to the three different components of the TACE software configuration shown in figure B1 (RTA, LVC Manager, and Test Network Manager). For this test event, all three specific test objectives were met.

It is important to clarify that the point of this test was not to perform a V&V of the TACE software, nor was it intended to satisfy an airworthiness process. Rather, this flight test was designed to demonstrate capabilities of the TACE system and to provide real world flight test data to compare with simulations of TACE.

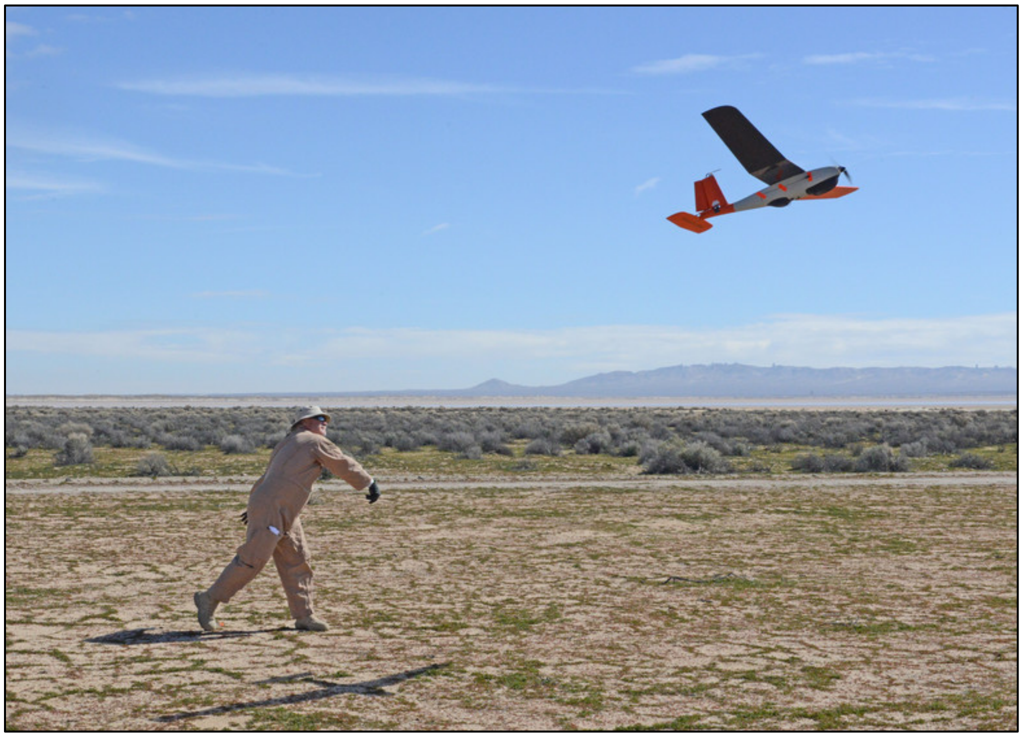

Figure B2 Takeoff of Lynx sUAS carrying TACE Payload (Feb 27 2019)

In July 2019, the same test cards were flown on the E-flite Shockwave aircraft. This group 2 UAS has a top speed of 200 knots and was chosen to demonstrate TACE's ability to replicate its successful results on a more dynamic aircraft. The success of TACE's RTA functionality on a high-speed aircraft builds confidence in the system and showcases the ability of TACE to accommodate an array of test aircraft.

FUTURE DEVELOPMENT AND USAGE

TACE is the current autonomy test capability of the 412 Test Wing at Edwards AFB. Furthermore, TACE is featuring prominently in several autonomy development programs in the DoD. Due to this fact, there is an ongoing effort to increase the capabilities of the TACE system, with a specific emphasis on improving the RTA functionality. There is active development ongoing to provide a RTA functionality for lower level control inputs (i.e., rate and surface level commands), incorporate a layered, smart remediation capability to account for more complex scenarios, and to include predictive avoidance functionality to avoid airspace incursions. The current development cycle features 6 weeks of TACE development followed by a week of flight test. This cycle is planned for the entirety of FY20. Beyond FY20, it is anticipated that TACE will continue to mature and be used for groups 1-3 UAS.

Current efforts with TACE are determining how to “certify” it as an approved autonomy test capability that can fly on a military range. “Certify” is a loaded word and means different things to different people. The burden for certifying a system to operate in the national airspace is much greater than operating in restricted airspace, away from the general population. Currently, the ET CTF is tracking total flight hours and number of RTA failures to help calculate the Mean Time Between Failure (MTBF). The MTBF parameter is failure ubiquitous within the range safety community and is a good metric for determining the trustworthiness of TACE for maintaining range and test safety.

The current test plan incorporates improvements and testing of TACE, but also leverages TACE to test autonomy software. As TACE matures and the path towards “certification” becomes more transparent, less additional safety mitigations will be required and a more robust autonomy test capability will be possible.

CONTACT INFORMATION

For all questions and comments concerning TACE, please contact Mr. Jeff Jessen, Chief Engineer of the Emerging Technologies CTF (email: jeffrey.jessen@us.af.mil, work: 661-275-1258)

COOPERATIVE OPERATIONS IN DENIED ENVIRONMENTS (CODE):

System Description:

The Collaborative Operations in a Denied Environment (CODE) autonomy system was developed under DARPA. It was designed to be an open, standards-aligned autonomy architecture enabling heterogeneous teams of UAVs to collaborate even in Comms/GPS denied environments. The CODE is a Software and Autonomy Open Architecture where the DARPA reference architecture interfaced with the Piccolo Autopilot. During DARPA CODE Tests and Extended TigerShark Experimentation (ETE) events, a JHU/APL-developed TACE-based software called White Force Network (WFN) was used as the LVC component. A high-level depiction of the CODE software architecture is presented in figure B3.

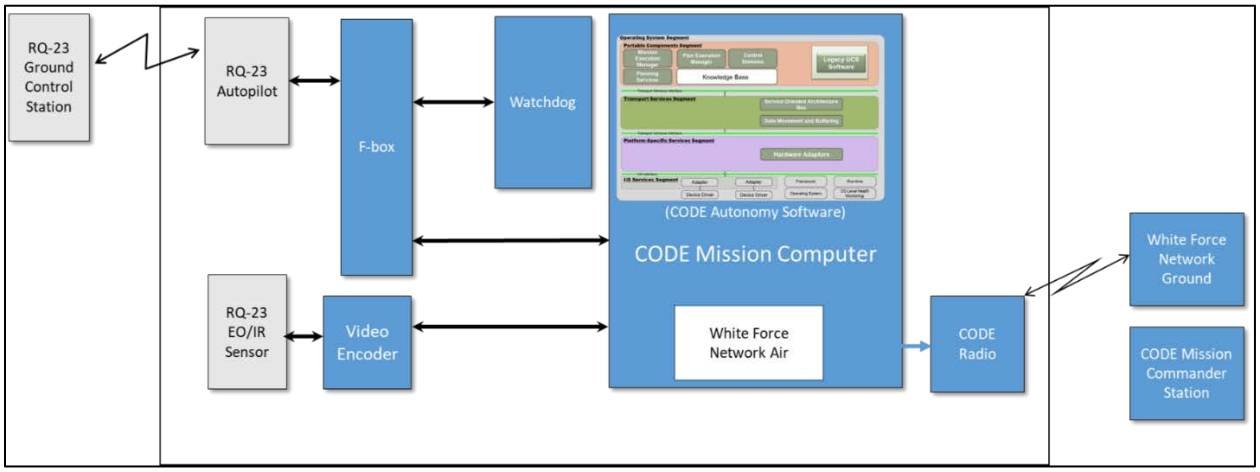

Figure B3 CODE Software Configuration

White Force Network (WFN) Air sits between the Autonomy and all interfaces (e.g., Watchdog, CODE Radio). The data flowing through the interfaces are manipulated in order to support test events. The WFN Ground was the software a Test Operator used to control how and what data were manipulated. An example of this manipulation was denying GPS to the CODE Autonomy. The operator commanded the WFN Ground to deny GPS. The WFN Air received the command and prevented all data from the Watchdog to the CODE Autonomy Software.

The Watchdog was a Raspberry Pi II single board microcomputer running a Linux operating system. The purpose of the Watchdog was to:

- Function as a message translator between the CODE Mission Computer's (CMC) STANAG 4586 message format and the TigerShark's Piccolo native datalink stream format.
- Provide message filtering to allow only a limited set of valid translated commands from the CMC through to the Piccolo autopilot.
- Provide control of CMC Level of Interoperability states through Air Vehicle (AV) Operator manipulation of the Watchdog Graphical User Interface.

SUMMARY OF FLIGHT TEST EXPERIENCE

<u>DARPA Code:</u>

Phase 1, consisting of Flight Test Series 1 (FTS1), was completed in April 2017. The FTS1 objective was to demonstrate and verify foundational CODE AV and Human-System Interface (HSI) functionality for a single vehicle through a series of tests that were designed to verify:

- Low-level task allocation through a "travelling salesman" test
- Bi-directional communications
- Basic common-operating picture sharing
- Mission authorization checks
- HSI single-vehicle play selection functionality
- HSI functionality

Phase 2 (FTS2) was completed in May 2017, with the objective of demonstrating multi-vehicle roles, multi-vehicle play selection and multi-vehicle behaviors through a series of tests that were designed to verify:

- Heterogeneous team play execution
- CODE system response to a detected threat
- The ability to assign vehicles different roles based on capability
- HSI multi-vehicle play selection functionality
- Simple target geo-location and sensor pointing (geo-locate virtually, point physically)
- Collaborative formation flying
- Coordinated time of arrival

Phase 3 concluded with the VIP Demo in February of 2019. The objective of Phase 3 was to validate systems capabilities, gain confidence in M&S results, and demonstrate collaborative autonomy to the greatest extent feasible onboard live UAV with associated challenges of operating onboard real hardware. Phase 3 included three distinct flight test campaigns and the final demonstration referred to as FTS3, FTS4, FTS5 and VIP Demo.

NAVY CODE RAIDER Extended TigerShark Experimentation (ETE):

The Navy ETE1 event occurred in June of 2019. Seven vignettes from the previous DARPA program were flown to demonstrate that an all-Navy Test Team could learn the skills needed to operate the software systems and conduct the event without CODE OEM support. 45 flight hours on 5 total real AVs (>6 Virtual AVs) were successfully flown during the 3-day period.

The Navy ETE2 occurred in September of 2019. ETE2 was a software regression test of CODE 2.1 (versus 1.7 that was flown for ETE1), and a newly developed Vignette (with Navy-built autonomy behaviors) designed to investigate CODE in a Navy specific scenario. Total flight time for ETE2 was 37 hours over 9 sorties, with 10 total Vignettes runs.

Future Development and Usage:

The CODE Autonomy architecture will be used again on other future Navy Autonomy experiments. The goal is to fly a new set of experiments once every six months.

CONTACT INFORMATION

For all questions and comments concerning CODE or the Navy's RAIDER Lab, please contact Mr. Charles Rea, Data Fusion Chief Engineer (email: charles.rea@navy.mil, work: (301) 342-9113)

APPENDIX C – UAS GROUPS

Table 1 DoD Definition of UAS Groups

UAS Group	Max Weight (lbs)	Nominal Operating Altitude	Speed (knots)
Group 1	<20	< 1,200 ft AGL	<100
Group 2	21-55	< 3,500 ft AGL	<250
Group 3	56 – 1,320	< FL 180	<250
Group 4	> 1,320	< FL 180	Any
Group 5	> 1,320	> FL 180	Any

This page intentionally left blank.

APPENDIX D – ABBREVIATIONS, ACRONYMS, AND SYMBOLS

Abbreviation	Definition
AAIT	Autonomy and Artificial Intelligence Testing
AF	Air Force
AFB	Air Force Base
AFI	Air Force Instruction
AFRL	Air Force Research Laboratory
AFSIM	Advanced Framework for Simulation, Integration, and Modelling
AFTC	Air Force Test Center
AGL	above ground level
AI	artificial intelligence
AIAA	American Institute of Aeronautics and Astronautics
API	application program interface
ARV	autonomous research vehicle
AV	air vehicle
CAL	critical abstraction layer
CAS	close air support
CMC	CODE Mission Computer
CMS	Cryptographic Message Syntax
CNSA	Commercial National Security Algorithm
CODE	Cooperative Operations in Denied Environments
COI	community of interest
CONOPS	concept of operations
CR	category ratio
CRPS	complacency potential rating scale
CSC	Conference on Scientific Computing
DARPA	Defense Advanced Research Projects Agency
DAU	Defense Acquisition University
DevOps	Development and Operations
DoD	Department of Defense
DoDD	Department of Defense Directive
DOE	design of experiments
DRM	design reference mission
DT	developmental test
DTIC	Defense Technical Information Center

Abbreviation	Definition
EO/IR	electro-optical/infrared
ET CTF	Emerging Technologies Combined Test Force
ETE	Extended TigerShark Experimentation
FL	flight level
ft	feet
FTS	flight test series
FY	fiscal year
GE	General Electric
GPS	global positioning system
HIL	hardware-in-the-loop
HRI	human robot interaction
HSI	human-systems integration
IDA	Institute for Defense Analyses
IEEE	Institute of Electrical and Electronics Engineers
ISR	Intelligence, Surveillance, and Reconnaissance
IT	information technology
ITEA	International Test and Evaluation Association
JHU/APL	Johns Hopkins University Applied Physics Lab
lbs	pounds
LVC	Live-Virtual-Constructive
M&S	modelling and simulation
MIT	Massachusetts Institute of Technology
ML	machine learning
MOE	measures of effectiveness
MOS	measures of suitability
MQT	mission qualification training
MTBF	mean time between failures
N/A	not applicable
OMS	Open Missions Systems
ORACLE	Operate Remote Aircraft Clairvoyantly in a Limited Evaluation
OSA	Open Systems Architecture
OT	operational test
RTA	Run Time Assurance
SA	situational awareness
SBTA	services-based testing of autonomy

<u>Abbreviation</u>	<u>Definition</u>
SDK	software development kit
SEAD	suppression of enemy aerial defense
SFG	synthetic force generator
STANAG	standardization agreement
STPA	systems theoretic process analysis
SUT	system under test
T&E	Test and Evaluation
TACE	Testing of Autonomy in Complex Environments
TEAMS	Teaming-Enabled Architectures for Manned-Unmanned Systems
TEVV	Test and Evaluation Verification and Validation
TIH	Technical Information Handbook
TRMC	Test Resource Management Center
TTP	tactics, techniques, and procedures
TW	Test Wing
UAS	unmanned aerial system
UAV	unmanned aerial vehicle
UCA	unsafe control action
U.S.	United States
USAF	United States Air Force
UxAS	Unmanned Systems Autonomy Services
V&V	verification & validation
VISTA	Variable In-flight Simulator Test Aircraft
VSS	Variable Stability Simulator
WFN	White Force Network
Wx	weather
>	greater than
<	less than

This page intentionally left blank.

APPENDIX E – DISTRIBUTION LIST

	Number of Copies	
Onsite	E-mail	Hardcopy
Emerging Technologies CTF Attn: 1st Lt Avery Leonard 759 North Base Rd Edwards AFB CA 93524 Email: avery.leonard.1@us.af.mil	1	0
Edwards AFB Technical Research Library Attn: Darrell Shiplett 307 E Popson Ave Edwards AFB CA 93524	0	2
AFTC/HO Attn: Jeannine Geiger 305 E Popson Ave Edwards AFB CA 93524 Email: jeannine.geiger@us.af.mil	1	0
Offsite		
Defense Technical Information Center 8725 John J. Kingman Rd, Ste 0944 Ft Belvoir, VA 22060-6217 Submit per DTIC procedures	1	0
Air Force Test Center Attn: Elisabetta L. Jerome, PhD, SL 101 West D Ave, Bldg 1, Suite 115 Eglin AFB 32542 elisabetta.jerome.1@us.af.mil	1	
Total:	4	2

This page intentionally left blank.