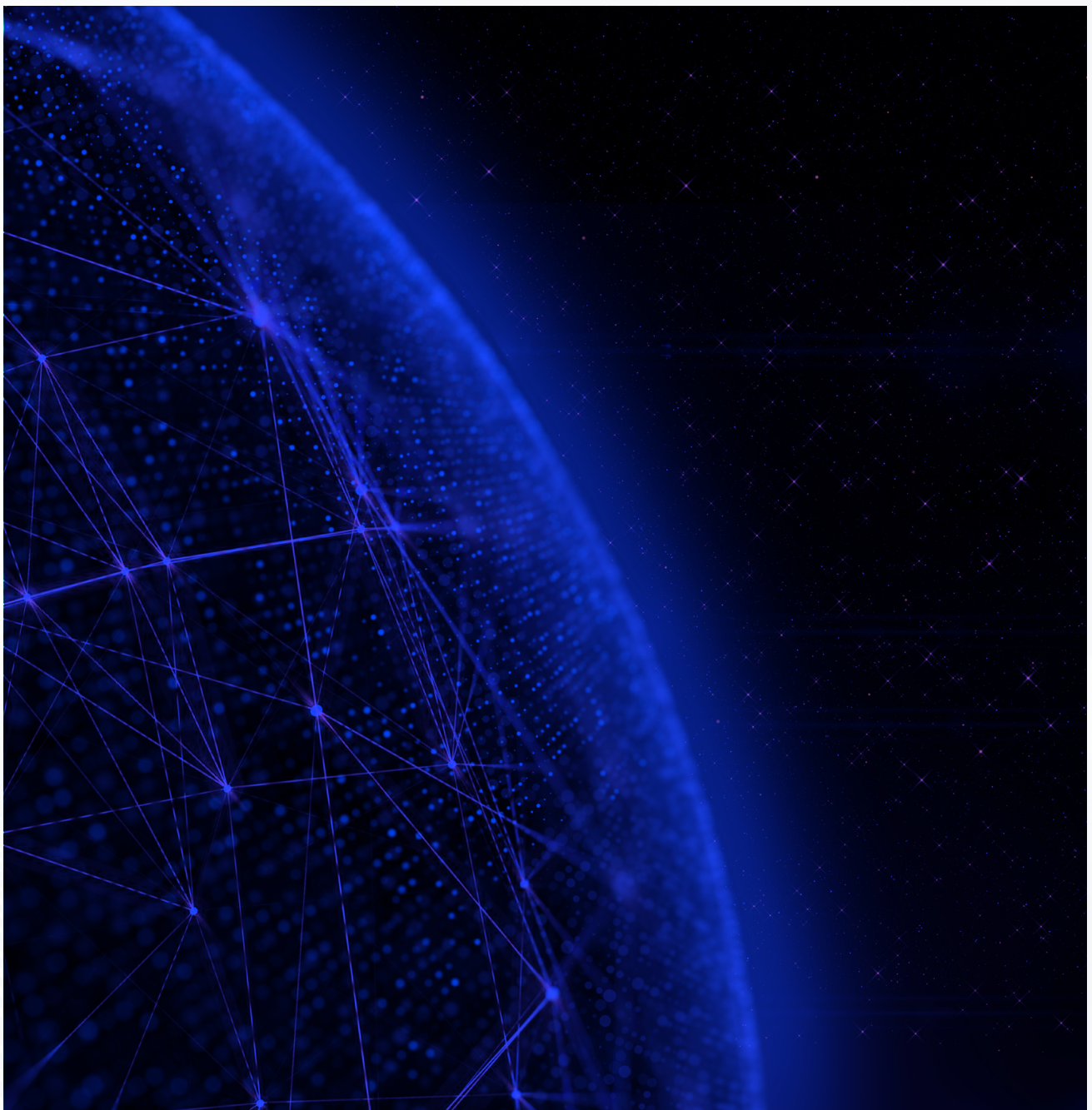


A Testing Framework

for E2E Zero Trust, Secure, Resilient, and Safe
Autonomous System Platform

Executive Summary

A fleet of zero-trust autonomous systems is a complex system with multiple components working together to achieve a common goal. With so many modular components involved in so many complex interactions, the overall system can malfunction for many reasons. Although researchers and vendors have explored and tested individual components, they still need to address the end-to-end security, resilience, and safety of fleets of systems, such as swarms or autonomous drones. This paper describes a rigorous, systematic testing framework that the Secure Systems Research Center (SSRC) of the Technology Innovation Institute (TII, Abu Dhabi, UAE) has evolved to evaluate complex systems and systems-of-systems—ensuring that the whole can cooperatively accomplish its missions.



Introduction

Autonomous zero-trust systems are doubly complex: they involve multiple multifaceted hardware and software components, and the underlying zero-trust architecture requires myriad security checks at every critical level of the system. Traditional approaches have focused on improving the resilience and security of individual autonomous agents—a car, a drone, a boat, a robot, etc. As we scale these up, however, the resultant fleets, swarms, and other huge, autonomous networks become exponentially more complex. [1]. Increasing complexity opens new doors for component-level and/or system-level failures, as well as increased attack surfaces for security breaches. In zero-trust autonomous systems, reliability and safety are paramount, particularly in critical-sector applications like defense, disaster response, and agriculture. Success demands a systematic and holistic testing framework ensuring a) that individual autonomous agents, their fleets, and the inter-agent interactions, as well as their human-interface interactions, all work as intended, and b) that such systems are robust and reliable.

A wide range of testing frameworks for autonomous systems [15, 2,3,5,6] have been developed, including a few for testing zero-trust systems [16,17]. However, these frameworks focus on limited aspects of the entire range of autonomous systems; they need to be improved if they are to accommodate comprehensive end-to-end testing. These frameworks must also be adapted to work with innovations in zero-trust autonomous technology.

For example, the Secure Systems Research Center (SSRC) at the Technology Innovation Institute (TII, Abu Dhabi, UAE) has been developing safe and secure tools for managing fleets of autonomous systems (e.g., unmanned aerial vehicles). Among these tools are the promising works on Ghaf, a zero-trust architecture for edge devices such as UAVs [9]; secure drone communication technology [10, 11]; and flight and mission operation tools for secure and safe swarm technology [13]. Building on the fundamental need for reliability, the SSRC is advancing the field by developing Secure Runtime Assurance (SRTA) protocols. This initiative aims to enhance safety and operational integrity in autonomous systems (particularly drones). By integrating AI and machine learning, SRTA addresses the inherent complexities of autonomous systems. It offers scalable, adaptive security solutions essential for meeting current regulatory challenges and anticipating future advancements [14]. Consequently, any testing framework must be tailored to accommodate emerging technologies.

Furthermore, the development framework must mirror the testing framework in its adherence to such regulatory requirements as:

- ISO/IEC 27001 for Information Security Management,
- ASTM F3201-16 for Small Unmanned Aircraft System (sUAS) Operations Over People,
- RTCA DO-178C for Airborne Systems and Equipment Certification.

Any effort to develop and test these technologies must also align with local regulatory requirements and comply with local regulatory authorities. For example, TII's SSRC must work with the UAE's General Civil Aviation Authority. Pilots testing the aerial systems must undergo comprehensive testing to meet local licensing requirements for flying aerial vehicles in public areas.

This whitepaper describes the systematic approach developed and adopted by SSRC to meticulously test fleets to assure they are secure, resilient, and safe. It reviews what is required to improve testing in each category, starting with the individual autonomous components (e.g., unmanned aerial vehicles) and scaling up to end-to-end testing for multiple unmanned traffic management systems and fleets of UAVs. This holistic testing approach includes reliability, performance and security across various levels and scales of zero-trust autonomous systems.

Although each type of autonomous agent can be used in many contexts, their behavior is similar when scaled into larger decentralized systems. In this context, it is useful to consider individual UAVs, swarms, robots, industrial controls, and cloud agents as nodes within a larger decentralized system.

This paper uses a fleet of UAVs to illustrate the holistic testing approach. However, many of the core principles and workflows described apply equally to fleets and swarms of unmanned ground vehicles (UGV), unmanned underwater vehicles, unmanned surface vehicles (boats), robots, and other autonomous things. Where appropriate, we refer to these collectively as UXVs. The same testing principles can also extend to other interactions with UXVs, including interactions by humans, industrial control systems, smart cities, and apps running on phones, workstations, and servers.

Some academic research has used the term “swarm” to describe collections of autonomous things working together. In this paper, we have adopted the term ‘fleet’ to characterize a similar concept more commonly used in an operational context.

Objectives of testing

The testing objectives for autonomous systems [3] are described below to ensure the successful, safe, and secure operation of fleets of autonomous UXVs.

1. Operational Functionality:

testing focuses on verifying that the UXV fleet performs its intended missions and tasks as designed. It assesses whether the drones can effectively communicate, coordinate, and collaborate, ensuring that their collective behavior aligns with the operational goals. It assesses the synchronization of flight patterns, the sharing of sensor data amongst the fleet, and the execution of complex tasks that require drones to make autonomous decisions based on dynamic data. Operational functionality tests also evaluate the fleet's ability to adapt to changes in the environment or mission parameters while maintaining high performance and consistency in task execution.

2. System Safety:

testing ensures that the system is operating safely. Since UXVs may possibly pose risks to themselves, humans, and safety-critical infrastructure, their safety needs to be tested carefully, particularly during regular system operation and failure modes.



3. System Resilience in Adverse and Extreme Conditions:

testing ensures that UXV fleets can maintain function in diverse and potentially adverse operational scenarios. These include potential disruptions from lousy weather, adversary intrusion, or system failures. The interdependence of fleet systems means the loss of a single UXV can severely impact the mission. Resilience testing includes assessing UXV fleets' performance against interferences or malicious attacks, and the fleet's ability to recover from such disruptions. A multistate network approach for resilience analysis can involve testing the fleet in various formations and conditions to evaluate how it adapts and self-organizes in response to changing environments.

4. Failsafe Mechanisms:

analysis and risk assessment are crucial for aviation safety, especially for UXVs operating in manned airspace and populated areas. This includes evaluating the fleet's responses to individual drone failures and the system's overall ability to maintain operational integrity in such events. The complexity of fleet behavior (due to the increased number of nodes) demands robust control systems that can handle failsafe mechanisms. Additionally, constraint-based formation and energy-consumption rates can affect failsafe protocols during real-world deployment testing, where the risk is inherently higher.

5. System Security:

testing of UXV fleets is essential to guard against malicious attacks, ensure safety and reliability, protect data privacy, and maintain user trust. These systems, which often perform critical functions, are vulnerable to cyber threats that can have serious consequences. Security testing can identify and address vulnerabilities, ensuring compliance with industry standards and regulations. This practice helps adapt to evolving cyber threats and mitigate insider risks, ensuring business continuity and the system's integrity.



System Description

A fleet of UXVs consists of multiple unmanned autonomous systems operating together to achieve a specific goal. The overall fleet system typically comprises hardware, communication systems, fleet management systems, and an autonomy software stack. These can vary across different classes of autonomous systems, such as UAVs, Unmanned Ground Vehicles (UGVs), robots, decentralized control systems, and autonomous cloud agents. As one example, the typical components found in a system of a fleet of UAVs are described below (Figures 1 and 2). Other types of systems typically have similar compositions.

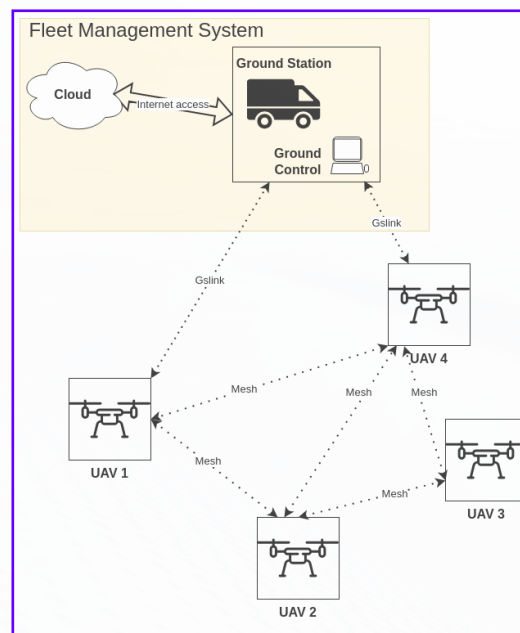


Figure 1: Typical components found in a fleet of autonomous UAVs (Dotted arrows: multicast UDP; Hollow arrow: Internet).

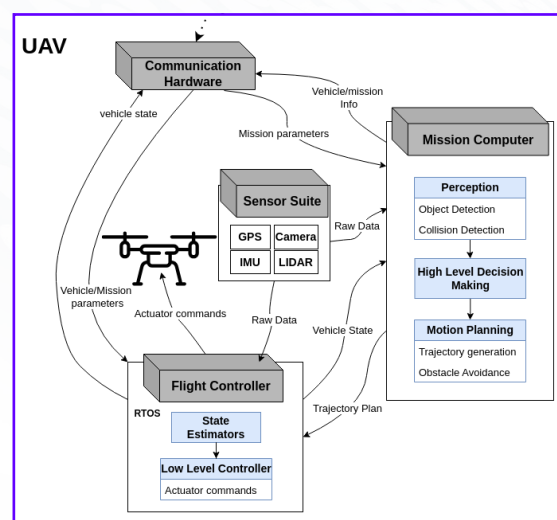


Figure 2: Typical components on an autonomous UAV (Solid arrows: Mavlink/XRCE-DDS protocols).

1. Hardware

A swarm or fleet of UXVs consists of multiple UXVs. The individual UXVs have sensors, communication hardware, and onboard processing units. To improve their function, resilience, safety, and security, UXVs use separate hardware components for flight control and mission-critical computation.

In particular, each UXV is equipped with dedicated flight-control hardware (flight controller), such as Pixhawk, which functions as the vehicle's brain. These controllers often have a Real-Time Operating System (RTOS) to handle real-time processing needs and ensure responsive and reliable flight control. The RTOS provides the necessary framework for these systems to execute time-critical tasks with high precision, essential for maintaining flight stability and safety.

In addition, each UXV features a separate mission computer that extends the flight controller's capabilities and provides advanced processing power for complex computations. This computer runs various modules in different programming languages, facilitating environment perception, advanced navigation, and strategic planning. It also manages internal communications and sensor data processing using frameworks like ROS2.

Apart from the individual UXVs, a ground control station serves as the system operator and fleet interface. Often, a cloud server communicates with the UXVs, ground control station, or both.

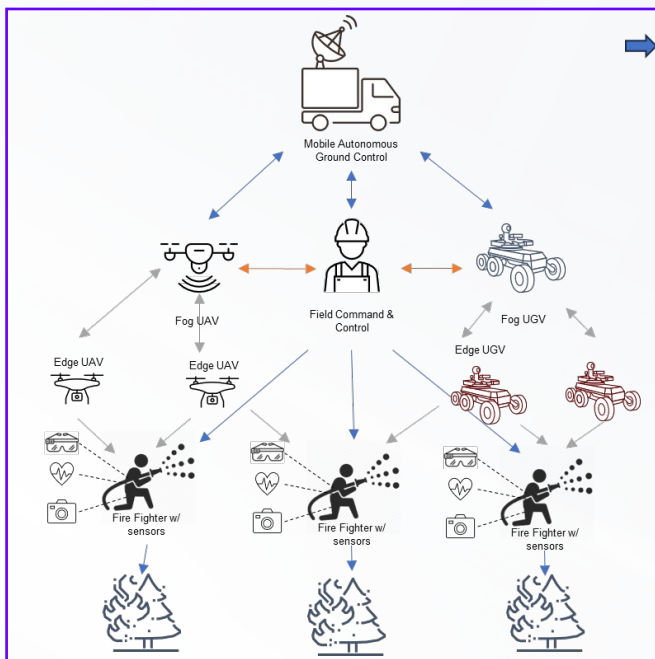


Figure 3: Different levels of connectivity in an autonomous fleet firefighting use case.

2. Communication Systems

Communication systems are fundamental to the efficacy of autonomous UXV fleets. They ensure secure and reliable data exchange within the fleet and between the fleet and its operator. The communication network facilitates the flow of control commands and telemetry and enables decentralized decision-making and self-organization, which are critical for overall fleet intelligence.

Each UXV and ground-control station has dedicated communication hardware to establish links with the rest of the entities in the system (See Figures 1 and 2).

The mission computer utilizes protocols such as multicast UDP (dotted arrow links in Figure 1) for efficient data exchange with UXVs or ground control stations for external communications. It employs MAVLink or Micro XRCE-DDS (solid arrow links in Figure 2) for reliable messaging between the flight controller and the mission computer. The UXVs and ground control stations interact with cloud-based services via the internet (hollow arrow link in Figure 1).

Various network topologies [4] can be used for communication, each with its own advantages (see an example system in Figure 3). Mesh networks, for example, are especially well-suited to the dynamic nature of drone fleets. This topology lets drones communicate directly with one other, creating a network with multiple redundant pathways for data transmission. This redundancy is critical in maintaining uninterrupted communication even if some drones experience failures or disruptions. This makes the mesh network a preferred choice for maintaining the high resilience and adaptability required for drone fleet operations.

3. Fleet Management System

The Fleet Management System plays a crucial role in the functionality of autonomous UXV fleets, serving as the central hub for managing and overseeing the fleet's complex operations—including the commissioning, provisioning, and intake of devices into the fleet. Ground or cloud-based control systems are integral to this system, providing real-time monitoring capabilities and the option for human intervention when necessary; at the same time, they also facilitate software management, OTA updates, and identity management. These systems offer sophisticated interfaces that enable operators to oversee flight paths, mission planning, and strategy execution, ensuring that the drones operate cohesively and accomplish their objectives efficiently. Software management capabilities ensure that drones are equipped with the latest software tailored to their tasks and operational needs. Over-the-air (OTA) updates enable seamless remote deployment of bug fixes, security patches, and new features. Identity management ensures secure authentication and authorization of both drones and operators within the fleet ecosystem.

Such systems facilitate direct control and allow integration of advanced algorithms that can autonomously optimize fleet behavior. For instance, adaptive real-time strategies let drone fleets navigate and complete missions in intricate environments, such as tracking objects in areas with dense structure or vegetation coverage or other visual impediments. Cloud-based control architectures further enhance this capability by enabling collaborative and simultaneous fleet control, allowing for a scalable and flexible approach to mission management across various scenarios.

These systems are vital in managing the intricacies of fleet coordination, from simple navigational tasks to complex adaptive mission execution. They are proving indispensable for autonomous UXV fleets' sustained adaptability and robust performance.



4. Autonomy Software Stack:

The autonomy software stack is a collection of algorithms that run on various system hardware components—e.g., flight controller, mission computer, ground control station, cloud, etc.—and ensures successful autonomous fleet operation [20]. In the intricate architecture of autonomous UXV fleets, these algorithms play pivotal roles at multiple levels, dictating the fleet’s collective behavior and each drone’s actions. At the highest level, the Fleet Management System, algorithms define the fleet’s strategic objectives and operational parameters, orchestrating the overall mission profile and ensuring that each UXV’s activities align with the broader goal.

For example, in a fleet with leader-follower configuration, the leader and follower UXVs may run different algorithms. A set of algorithms running on the leader’s mission computer will translate the fleet management system’s overarching mission strategy into tactical maneuvers. The leader’s instructions maintain formation integrity, allow dynamic rerouting, and facilitate obstacle avoidance, ensuring that the fleet’s collective behavior adapts to real-time environmental feedback.

Aboard the follower UXV, algorithms are critical for interpreting and enacting the fleet leader’s commands. These algorithms manage the UXV’s local-environment perception, motion plans, and immediate reactions to unforeseen changes or threats. These algorithms must be finely tuned so that each UXV can operate as part of the collective under the leader’s direction, or autonomously if it loses communication with the fleet. For a detailed description of a typical software system on a UAV, please refer to the “PX4 Architectural Overview.” [8].

Embedded within these algorithmic layers are failsafe mechanisms designed to preserve the mission’s integrity and the UXVs’ safety. At the fleet management level, failsafes ensure that corrective actions are initiated if the fleet deviates from mission parameters. For leader UXVs, failsafes might involve alternative planning to reorganize the fleet in the event of disruptions. The follower UXVs have their own failsafes to handle scenarios like GPS signal loss or battery failure, allowing for autonomous emergency landing or return-to-home procedures.

These algorithms, functioning at levels from macro-operations to micro-level controls, are not standalone; they are interdependent, creating a robust system that ensures the fleet is resilient, adaptive, and capable of sustaining its operations under various conditions.

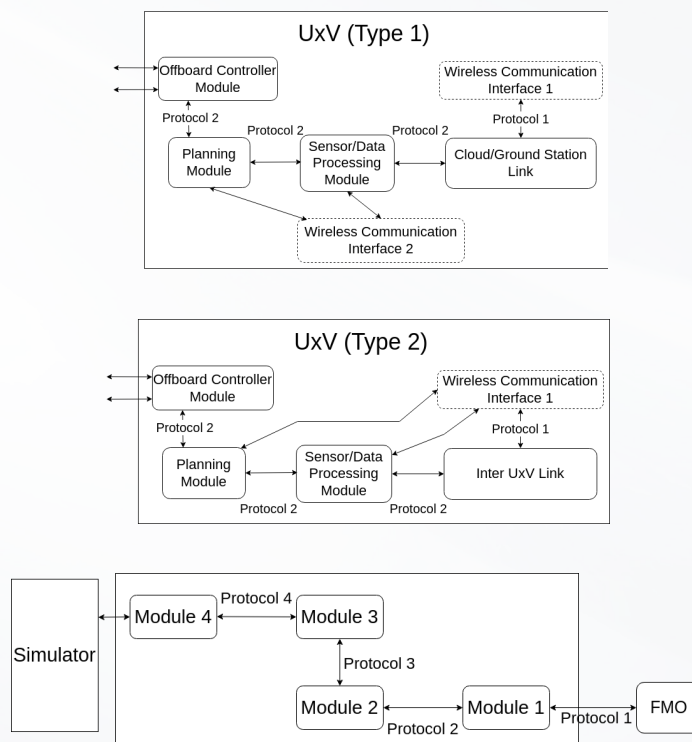


Figure 4: Interconnection of essential components in an autonomous fleet.

Testing Framework

At SSRC TII, the elaborate testing framework is divided into development phases to help identify and rectify issues most efficiently and cost-effectively at each stage. The SSRC approach breaks the testing process into four phases: 1) unit testing, 2) integration testing with software in the loop, 3) integration testing with hardware in the loop, and 4) real-world testing. The essential elements of these stages are described below.

Phase-1: Unit Testing

As the name suggests, unit testing refers to testing every single component of the system separately, independent of the rest of the system. Unit testing is critical in establishing individual components' reliability and operational safety and, in turn, the overall system. Through unit testing, developers ensure that each element functions in isolation; the elements include intricate software algorithms, communication protocols, and hardware such as sensors and flight actuators. Any faults overlooked at the component level could easily penetrate the system, potentially making the system unsafe and unreliable. For example, a faulty sensor or actuator could generate unintended behavior, making it dangerous to operate. During the unit-testing phase, all sensors and actuators should be tested in isolation to identify damage or faults.

Overall, the meticulous scrutiny provided by unit tests lets developers detect and correct flaws early, safeguarding the fleet's security and operational integrity. This essential step facilitates smoother integration and lays the groundwork for robust safety measures. Drone fleets, often deployed in dynamic and uncertain environments, benefit immensely from the durability that comes with intensive unit testing. This ensures that each drone is capable of independently managing unexpected conditions and contributing to the fleet's overall mission success.

Far-reaching unit testing (in the context of secure autonomous drones) reinforces the stepwise assembly of systems designed for high levels of autonomy and dependability. Rigorous component validation through unit testing is fundamental to fostering confidence in these sophisticated systems, which are destined to transform various industries by accomplishing tasks previously deemed impractical for automated solutions.

However well tested individual components/modules may be during Phase 1, their overall interaction with other system elements—and the system's overall performance, safety, reliability, and security— cannot be evaluated by unit testing alone. That is where integration testing becomes vital.

Phase-2: Integration Testing with a Software-in-the-Loop Digital Twin

During the integration testing phase, individual components validated by unit testing are combined and tested as a group. This pivotal phase reveals issues that may not be apparent when components are tested in isolation, issues such as data-flow problems, interface mismatches, and unexpected interactions between software and hardware modules.

This phase uses a software-in-the-loop (SIL) digital twin, a precise virtual simulation of the drone's real-world operations. This approach allows thorough testing of the integrated system's responses to virtual scenarios that could be unsafe, costly, or impractical to replicate in the real world. It provides an opportunity to stress-test the system under a range of operational conditions that includes adverse weather, signal jamming, and unexpected obstacles. Using a digital twin in the loop, developers can verify and refine interactions among the UXV's navigation, collision avoidance, communication protocols, and payload management systems, all in a controlled yet realistic setting.

The SIL digital twin replaces every system component by an equivalent software entity that faithfully reproduces the behavior of the original component. These equivalent software components interact with each other via different middleware—on the same computer or on multiple computers on the same shared network. (See Figure 5 for the components of a typical SIL system.)

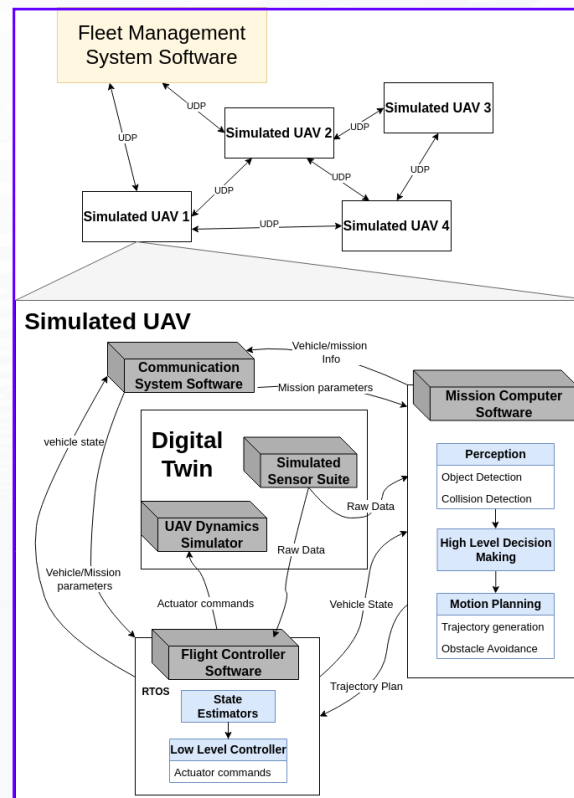


Figure 5: Software in the Loop (SIL) Architecture.

SIL Testing Methodology

The overall SIL software stack is a collection of multiple software components (called modules) that interact with each other at multiple levels. We propose a bottom-up approach that begins with the last module in the workflow sequence and progressively incorporates each preceding module, one by one (see Figure 6). Here is a structured outline of the methodology:

1. Initial Stage (Module n):

- Begin with the final software module, which directly interfaces with the simulator.
- Trigger a mission or command from this module to verify the outcome in the simulator's test interface. The simulator serves as the ground truth reference at this stage.
- Ensure that the simulator accurately reflects the command's intended effects and confirm that module n operates correctly within the simulated environment.

2. Intermediate Stages (Modules n-1 to 2):

- Proceed to the second-to-last module (module n-1), initiating the mission or command from here.
- At each stage, the interaction is verified at two levels:
 - The test interface of the module that initiates the command (to confirm that it is sending the correct command down the line).
 - The test interface of the module that receives the command (to confirm that it has correctly received and processed the command).
- This dual verification is carried out progressively, moving from module n-1 to module 2, each time integrating the next module in the sequence and ensuring the protocols interact as expected.

3. Final Stage (Module 1):

- The first module triggers the mission or command, which is verified across all subsequent modules' test interfaces and the simulator.
- This comprehensive testing confirms the integrity of the entire software stack and the proper flow of information through all protocols.

4. Fleet Integration Stage:

- In a simulated environment, deploy multiple drones, each equipped with the fully integrated software stack.
- Use the digital twin simulation to execute the same bottom-up testing methodology, now at the fleet level.
- Verify both the intra-drone software interactions and inter-drone communications and behaviors.
- The digital twin simulation will provide a complex, dynamic environment to evaluate the fleet's collective response to commands.

5. Scenario Based Testing:

- At this stage, testers create a variety of scenarios for the entire fleet system. They observe and evaluate each scenario. There are countless theoretical scenarios, but in practice, priority goes to testing scenarios that specifically target critical features of the system. For example, validators might test navigation scenarios to confirm that collision-avoidance modules work as intended, or they might test scenarios to verify that the fleet is capable of achieving its goals in different environmental conditions, etc.

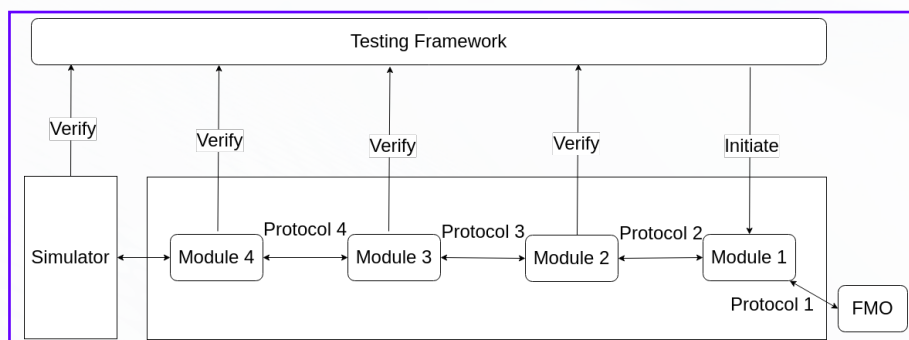


Figure 6: A representative Software in the Loop testing flow.

This testing methodology ensures that each module and its corresponding protocol are individually and collectively validated in a controlled, sequential manner. It provides a meticulous framework for capturing and addressing the intricacies and nuances of the UXV's software systems. This approach methodically builds confidence in the UXV's capabilities by testing from the smallest unit (the individual module) to the complete system (a fleet of UXVs). It ensures that the integrated fleet behaves as intended when deployed in a digital-twin environment. In summary, Phase 2 Integration software-in-the-loop digital-twin testing of the autonomous fleet extends beyond the scope of Phase 1 unit testing by focusing on the interactions and interoperability among the system's modules and components. While Phase 1 Unit Testing verifies the functionality of individual units or components in isolation (ensuring that each part works as specified), Phase 2 Integration Testing takes a more holistic approach. It assesses how well the integrated components work together, simulating real-world scenarios and use cases that the secure autonomous fleet is expected to encounter.

Phase-3: Integration Testing with a Hardware-in-the-Loop Digital Twin

After SIL integration testing comes Hardware-in-the-Loop (HIL) testing using a digital twin. By integrating physical components into the testing environment, this HIL phase provides important added benefits and critical information.

1. Hardware Execution and Performance:

- HIL testing validates the hardware's ability to execute instructions under real-world operating conditions. Thus, designers can assess performance of key components like processing units as they are subjected to the actual stresses and strains, they will encounter during operation.
- Each hardware component—the various Systems on a Chip (SoCs) and architectures such as x86, ARM, and RISC-V within autonomous systems—brings its unique characteristics, performance profiles, and potential challenges. HIL digital-twin testing reveals their nuances and operational issues.
- HIL tests give designers an opportunity to observe how the hardware interacts with the software in real-time, helping them identify discrepancies that may arise when transitioning from a virtual to a physical environment.

2. Realistic Communication Latencies:

- Unlike SIL, in which communication is instantaneous, HIL testing introduces the real-world latencies associated with hardware and wireless interfaces. This allows testers to evaluate communication protocols and algorithms over the physical radio layer under realistic network conditions, ensuring the system can maintain performance and reliability despite potential delays.

3. Power Consumption and Thermal Dynamics:

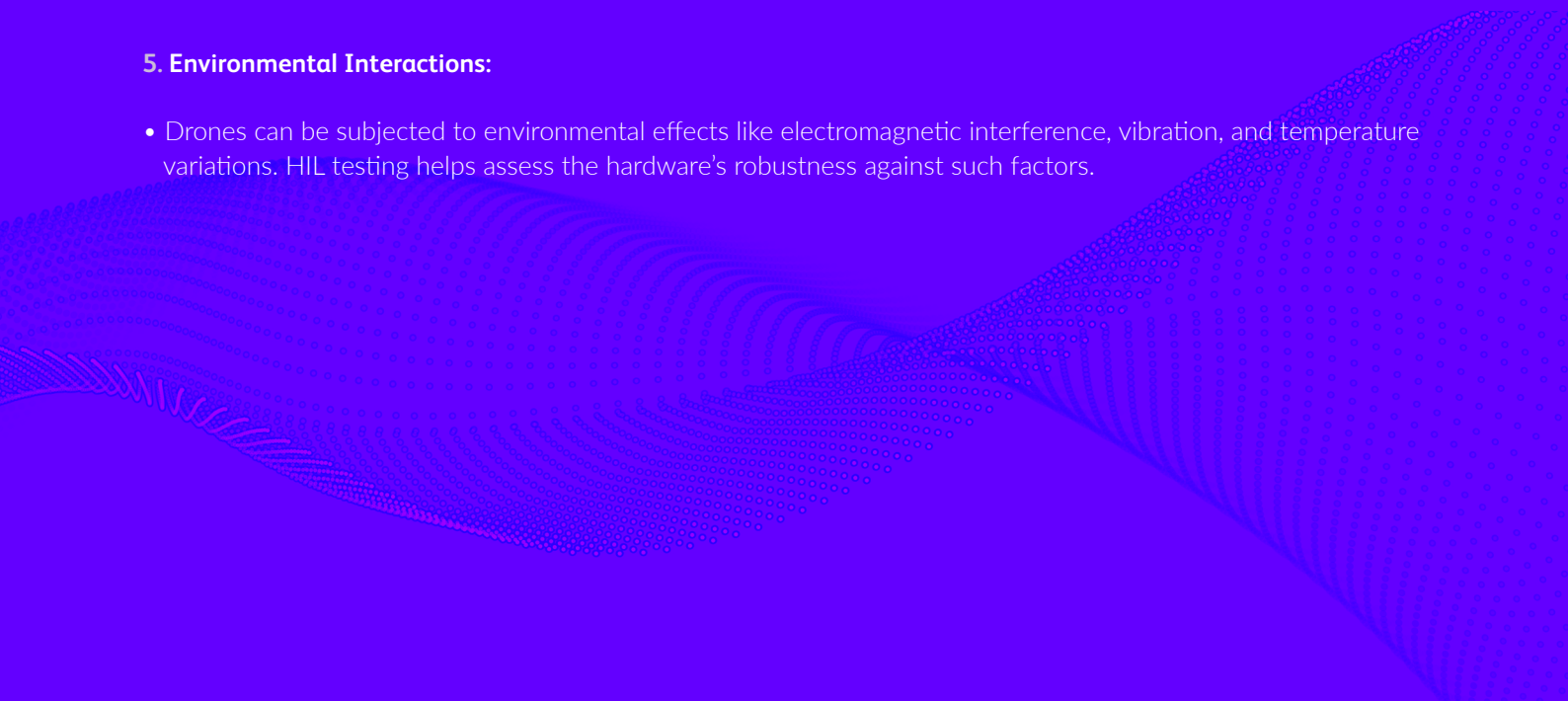
- HIL testing can expose the hardware's power consumption patterns and thermal characteristics under operational loads. This is crucial for optimizing energy efficiency and ensuring that the hardware can dissipate heat effectively during extended missions.

4. Sensor and Actuator Fidelity:

- This phase confirms the accuracy and reliability of sensors and actuators. It is vital for certifying that the drone's perception and interaction with the physical world align with expectations.

5. Environmental Interactions:

- Drones can be subjected to environmental effects like electromagnetic interference, vibration, and temperature variations. HIL testing helps assess the hardware's robustness against such factors.



HIL Testing Methodology

Here is a simple outline of SSRC's proposed HIL testing methodology:

- 1. Integration with Digital Twin:** This process begins by integrating the physical hardware components with the digital twin simulation. This hybrid setup mirrors the drone's operation in a virtual environment, but employs actual hardware.
- 2. Sequential Component Testing:** Similar to the SIL methodology, designers initiate testing with the final module and progressively integrate each preceding module, but this time with actual hardware components in the loop.
- 3. Communication and Latency:** This step focuses on the communication stack to guarantee that data transfer between hardware and software components is resilient enough to cope with the latencies observed in real-world wireless communications.
- 4. Sensor and Actuator Verification:** Evaluators conduct tests of sensor accuracy and actuator response times, validating these against the digital twin's predicted outcomes.
- 5. Power and Thermal Testing:** These tests monitor power consumption and thermal output under simulated mission scenarios to identify issues that could affect mission endurance and hardware reliability.
- 6. Fleet Dynamics:** Finally, testers scale up to a fleet environment, in which multiple drones (each with its own hardware integrated into the loop) interact within the digital twin simulation. This tests the fleet's collective hardware performance and ability to execute coordinated tasks.

HIL testing can help developers comprehensively understand the drones' operational readiness, bridging the gap between theoretical models and actual deployment conditions. This phase is essential for ensuring that the autonomous drones will perform reliably and securely as a cohesive unit within a fleet when it takes to the skies.

Phase-4: Real World Testing

Real-world testing for UXVs and fleets of UXVs splits into three parts conducted in various environments. Tests at each level focus on safety, so that there is always a safety pilot who can take control of the drone and perform a safe landing if the autonomous system is not performing as it should.

Basic testing: Here, we check the drone system's basic functionality, including its flight controller, mission computer, and communication system. It is enough to see that the systems are working (i.e., haven't failed), and the fleet is able to execute quick missions while all components remain stable.

Intermediate testing: At the next step, we begin to stress-test the system to see how it reacts during operation. Tests at this level can include high-speed tests, real-time mission updates, checks of communication reliability across the fleet, etc. **Reliability testing:** In the next step, we repeat all the tests initially carried out at the intermediate level, but focus on how many times the system can execute the test cases at the intermediate level before failure.

Resiliency testing: Finally, we check whether the system can recover from failures or can operate normally despite adverse conditions or component-level or system-level failures. For example, can a UXV operate normally even if one or more of its actuators fails? Evaluators will devise specific resiliency tests to reflect UXV mission requirements and operational challenges.

Real-World Test Environments

Real-world testing should be conducted in a series of progressively more challenging environments.

- **Laboratory Testing:** Initial real-world testing is conducted in a controlled setting with emulated network conditions and artificial interference. For SSRC, this is an indoor flight laboratory where we use the output of motion capture cameras to mimic actual GPS coordinates. Since GPS coordinates are unreliable indoors, the drone can use this to navigate effectively. In this controlled environment, we can simulate the drone's reaction in different emergency situations without posing any threat to the immediate environment and without having to worry about the effect of environmental elements like wind, heat, and sunlight.
- **Controlled Outdoor Testing:** After lab testing concludes, the next tests move to more realistic environments with controlled interference. Testing occurs in an outdoor, net-covered cage. The outdoor location ensures access to GPS signals while exposing the UXVs to environmental elements. The cage ensures the UXVs remain in the test area if an emergency occurs.
- **Field Testing:** Final real-world testing puts the UXVs in selected scenarios under close observation. Tests are conducted in open-air areas that fit the selected scenarios and match the conditions in which the fleet of UXVs would actually operate.



Security Testing

Security testing, in combination with drone system testing, ensures the robustness and reliability of the system in the face of potential security threats. In the context of zero-trust autonomous systems, a zero-trust model assumes that no entity, whether inside or outside the organization, should be trusted by default. Thus, security testing must investigate potential gaps. Here are the objectives of security testing in zero-trust autonomous systems projects:

Objectives:

- Discover and assess vulnerabilities in the system's architecture, components, and communication channels.
- Verify the effectiveness of access controls and authentication mechanisms.
- Evaluate the methods in place for encrypting, storing, and transmitting sensitive data.
- Monitor and analyze how the autonomous system detects anomalies and potential security threats.
- Assess the security of communication channels to ensure confidentiality and integrity of data in transit.
- Evaluate the security posture of software components, including algorithms, protocols, and applications.
- Implement continuous monitoring mechanisms to detect and respond to emerging security threats.
- Validate that the autonomous system complies with relevant security standards and regulations.
- Evaluate the implementation of mitigation strategies for known vulnerabilities.

Various methods and techniques can be employed to attain these objectives. They help identify vulnerabilities, assess access controls, protect data, analyze behavior, and ensure the system's overall security. Common security testing approaches include:

- **Penetration Testing:** Identify vulnerabilities in the system by simulating real-world attacks and validate implemented mitigation strategies.
- **Security Audits:** Assess the overall security posture of the system.
- **Code Review:** Ensure the security of the software components.
- **Vulnerability Scanning:** Identify and assess vulnerabilities in the system's infrastructure.
- **Compliance Assessments:** Validate compliance with security standards and regulations.
- **Continuous Monitoring:** Detect and respond to emerging security threats.

Zero-trust autonomous systems projects must particularly emphasize penetration testing. Penetration testing replicates real-world scenarios to provide a unique opportunity for unearthing the "unknown unknowns." By adopting an adversarial perspective and simulating attacks on the system, we can proactively fortify our security posture by strategically positioning ourselves as the assailants to uncover potential vulnerabilities that may otherwise remain undiscovered.

Owing to this importance, SSRC has meticulously delineated the scope of our penetration testing efforts. This strategic categorization helps enhance our campaign visualization, facilitates comprehensive coverage, and ensures that each facet of our operations receives meticulous attention.

Penetration Testing categories:

- **Communications:**

This category covers test cases dedicated to vehicle communications, spanning internal communication between software modules, inter-vehicle communication, and communication between the vehicle and the infrastructure. Examples of sub-categories include:

- Intra-vehicle
- Inter-vehicle
- Vehicle-to-infrastructure

- **Trusted Execution:**

This category focuses on deployed artifacts like binaries, container images, scripts, libraries, and firmware. Tests must affirm that executed binaries come from an approved whitelist or validate them via trusted methods like signatures or specific hashes (IMA). Examples of sub-categories include:

- Secure Boot
- Kernel Protection
- User-space Protection
- Trusted Updates

- **Software Hygiene:**

This category comprises security features designed to diminish the impact of an exploit and make it harder for attackers to compromise the system. These certify the current state of other software components, such as libraries, to prevent exploitable vulnerabilities. Examples of sub-categories include:

- Security Patches
- Run-time arguments
- Compile flags

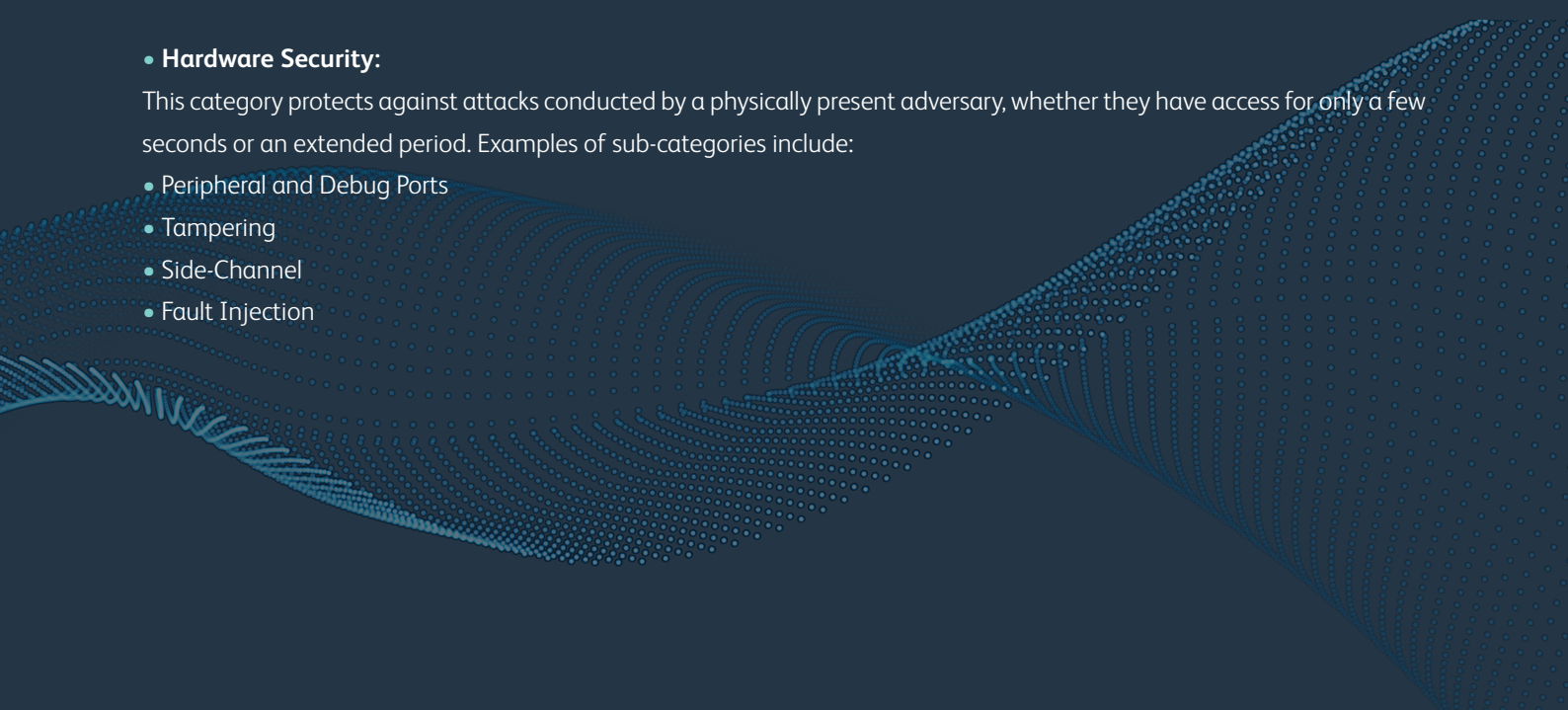
- **Local Data Protection:**

This category safeguards against attackers attempting to extract sensitive information from a device, either remotely or physically. Examples of sub-categories include:

- Access Rights
- Sensor Data
- Secure Storage

- **Hardware Security:**

This category protects against attacks conducted by a physically present adversary, whether they have access for only a few seconds or an extended period. Examples of sub-categories include:

- Peripheral and Debug Ports
 - Tampering
 - Side-Channel
 - Fault Injection
- 

Ongoing work

The framework presented here illustrates one approach developed by the SSRC, an approach addresses many limitations of current verification and validation frameworks, particularly when they are extended to systems of systems. Many challenges still remain and need to be addressed in the future.

Interference and jamming can play havoc with individual UXVs and fleets [5]. Testing frameworks must evolve to ensure reliable communication in the presence of accidental interference or deliberate jamming.

Testing must be scalable and adaptable to the needs of different types of systems. This will require making it easier to adapt the testing framework to fleets of varying sizes and complexities.

Developing a framework that can ensure autonomy in varying conditions is also important [6,7]. Further work on building digital twins that mimic more edge cases and additional adverse conditions will facilitate reliable autonomous operations in every anticipated operating environment.

It is also critical to consider the human factors in operating UXV fleets. This will be critical in use cases like firefighting and search-and-rescue, which require that humans collaborate with UXV fleets. More comprehensive user experience testing could help increase the acceptance and trust of people collaborating with UXV fleets.

The method-development phases outlined here are powerful. They have the potential to substantively improve all projects undertaken by the SSRC. For instance, our Secure Runtime Assurance (SRTA) initiative might benefit by employing the four-phase approach to evaluate the proposed machinery [14]. This evaluation could involve the direct injection of faults into the flight controller—in both the software and hardware in the loop—facilitating a rigorous assessment of how unexpected behaviors influence mission outcomes. Furthermore, there is an opportunity to refine machine learning models significantly by integrating real-world data as it comes in. This approach enhances the accuracy and reliability of these models and ensures that the SRTA project remains at the forefront of technological advancements in autonomous systems safety and operational integrity.

The TII's SSRC has also developed a novel research and development program for building secure autonomous fleets and testing their various failure modes. At its core, this effort includes research on the secure Ghaf Platform that leverages zero-trust principles to secure its central compute elements [9]. Today, SSRC is extending this core architecture to both workstations and UXVs. Support for other types of embedded systems is planned. Complementary work focuses on a zero-trust architecture to combat wireless network attacks [10]. Additional research explores how to scale security to autonomous fleets [11], whilst other works are exploring how to extend autonomic computing principles to systems of drone fleets to build safe, secure, and trusted systems-of-systems [12]. All of these efforts benefit from the parallel work described here to develop a more comprehensive verification and validation framework to test the safety and security of these complex systems.

In recent years, artificial intelligence (AI) and machine learning (ML) techniques have been deployed in autonomous systems to make them more intelligent, efficient, and adaptable [18]. While AI and ML techniques do bring these benefits to autonomous systems, their outputs are non-deterministic: testing them is a challenge [19]. Remember that zero-trust autonomous systems are often safety-critical, and current AI and ML tools remain prone to hallucinations and other forms of erroneous output. Our current and future goal is to incorporate testing of autonomous systems' AI and ML performance into the end-to-end testing protocol.

Conclusion

Rigorous testing of autonomous fleet drone systems is critical for their integration into commercial and public domains. The framework introduced here reflects some of the lessons SSRC has learned while developing guidelines for validating these systems' security, resilience, and safety. Continuous updates to these testing protocols will be required to keep pace with technological advancements and evolving threat landscapes.

It is also important to emphasize that the approach outlined here guide other organizations developing and deploying autonomous drone fleets. Implementing a testing program requires detailed planning, curating technical resources, and collaborating across disciplines to fine-tune the program for a given set of objectives.

At a high level, we recommend that teams consider several essential elements in developing these programs. First, an iterative testing program is required to tune the framework for each use case. Teams will also need to consider regular updates to testing protocols, based on the latest threat assessments and technological developments.

Second, cross-sector collaboration is essential. Autonomous systems-of-systems are highly complex and can sometimes affect stakeholders in non-intuitive ways. Engagement with industry, academia, and regulatory bodies can help standardize testing best practices that depend on different types of expertise.

Third, as an industry, we are still in the early phases of trying to fully understand how autonomous systems can fail. Ongoing investment in R&D will be required to improve autonomous system testing methodologies.

Additionally, this investment can help to fine-tune testing methodologies across different types of autonomous systems, including drones, cars, robots, industrial controls, and cloud infrastructure. Different use cases may well have very different needs. At the same time, however, we have developed, and will continue to develop, general principles and architectures that will inform adjacent use cases.

References

- [1] M. Abdelkader, S. Güler, H. Jaleel, and J. S. Shamma, "Aerial Swarms: Recent Applications and Challenges," *Curr Robot Rep*, vol. 2, no. 3, pp. 309–320, Sep. 2021, doi: 10.1007/s43154-021-00063-4.
- [2] E. Zaitseva, V. Levashenko, R. Mukhamediev, N. Brinzei, A. Kovalenko, A. Symagulov, "Review of Reliability Assessment Methods of Drone Swarm (Fleet) and a New Importance Evaluation Based Method of Drone Swarm Structure Analysis". *Mathematics* 2023, 11, 2551. <https://doi.org/10.3390/math11112551> Available: <https://www.mdpi.com/2227-7390/11/11/2551>
- [3] Linz, Tilo. (2020). Testing Autonomous Systems. The Future of Software Quality Assurance (pp. 61-75) doi: 10.1007/978-3-030-29509-7_5.
- [4] G. Asaamoning, P. Mendes, D. Rosário, and E. Cerqueira, "Drone Swarms as Networked Control Systems by Integration of Networking and Computing," *Sensors (Basel)*, vol. 21, no. 8, p. 2642, Apr. 2021, doi: 10.3390/s21082642.
- [5] P. Zhang, T. Wu, R. Cao, Z. Li, J. Xu, "UAV Swarm Resilience Assessment Considering Load Balancing," *Frontiers in Physics*, vol. 10 no. 15, 2022, [Online]. Available: <https://www.frontiersin.org/articles/10.3389/fphy.2022.821321/full>
- [6] T. Liu, G. Bai, J. Tao, Y.-A. Zhang, and Y. Fang, "A Multistate Network Approach for Resilience Analysis of UAV Swarm considering Information Exchange Capacity," *Reliability Engineering & System Safety*, vol. 241, p. 109606, Jan. 2024, doi: 10.1016/j.ress.2023.109606.
- [7] A. Phadke and F. A. Medrano, "Towards Resilient UAV Swarms—A Breakdown of Resiliency Requirements in UAV Swarms," *Drones*, vol. 6, no. 11, Art. no. 11, Nov. 2022, doi: 10.3390/math11112551
- [8] "PX4 Architectural Overview | PX4 User Guide (main)." Accessed: Dec. 12, 2023. [Online]. Available: <https://docs.px4.io/main/en/concept/architecture.html>
- [9] Secure Systems Research Center, Technology Innovation Institute, "Ghaf Compute Platform Virtualization on the Edge", Accessed: Dec. 16, 2024. [Online]. Available: https://files-prod.tii.ae/static/publications/IO-230049-SSRC-GHAF-Whitepaper_V9.pdf
- [10] Secure Systems Research Center, Technology Innovation Institute, "Secure Comms Shield: A Zero-Trust Network Solution for Distributed Communications" Accessed: Dec. 16, 2024. [Online]. Available: https://files-prod.tii.ae/static/publications/IO-220140-TII_SSRC_Secure_Comms_White_Paper_FINAL.pdf
- [11] Secure Systems Research Center, Technology Innovation Institute, "Building a Zero Trust Security Model for Autonomous Systems." *IEEE Spectrum*, Accessed: Dec. 12, 2023. [Online]. Available: <https://spectrum.ieee.org/zero-trust-security-autonomous-systems>
- [12] K. AL-Dosari, Z. Hunaiti, and W. Balachandran, "Systematic Review on Civilian Drones in Safety and Security Applications," *Drones*, vol. 7, no. 3, Art. no. 3, Mar. 2023, doi: 10.3390/drones7030210.
- [13] Secure Systems Research Center, Technology Innovation Institute, "A Zero Trust approach to autonomous systems of systems security." *IEEE Spectrum*, Accessed: Dec. 16, 2024. [Online]. Available: https://files-prod.tii.ae/static/publications/IO-220021-TII_Autonomous_Security_White_Paper_V4-16042022%5B79%5D.pdf
- [14] Secure Systems Research Center, Technology Innovation Institute, "Discover the Game-Changing Potential of Autonomous Systems with SRTA." *IEEE Spectrum*, Accessed: Apr. 01, 2024. [Online]. Available: <https://files-prod.tii.ae/static/publications/IO-230180-Building-Safer-Collectives-of-Autonomous-Robots-with-Secure-Runtime-Assurance-SRTA-1March2024.pdf>
- [15] Q. Song, E. Engström, and P. Runeson, "Concepts in testing of autonomous systems: Academic literature and industry practice" *IEEE/ACM 1st Workshop on AI Engineering-Software Engineering for AI (WAIN)* (pp. 74-81), May 2021.
- [16] D. Vincenzo, and W. R. Michalson. "A Survey on Trust Metrics for Autonomous Robotic Systems." *arXiv preprint arXiv:2106.15015* (2021).
- [17] P. Gari, A. Selwyn, and D. Zwillinger. "The "Trust V": building and measuring trust in autonomous systems." *Robust Intelligence and Trust in Autonomous Systems* (2016): 55-77.
- [18] M. Soori, B. Arezoo, and R. Dastres. "Artificial intelligence, machine learning and deep learning in advanced robotics, a review." *Cognitive Robotics* (2023).
- [19] Wu, Tingting, et al. "Testing Artificial Intelligence System Towards Safety and Robustness: State of the Art." *IAENG International Journal of Computer Science* 47.3 (2020).
- [20] A. O. Prasad et al., "Design and development of software stack of an autonomous vehicle using robot operating system," *Robotics and Autonomous Systems*, vol. 161, p. 104340, Mar. 2023, doi: 10.1016/j.robot.2022.104340.