Optimizing Defense AI with Simulation-Driven CI/CD

Victoria Dorn, Andres Ulloa, Andrew Snedeker, Anastacia MacAllister, Rey Nicolas

General Atomics

Poway, CA

Victoria.Dorn@ga-asi.com, Andres.Ulloa@ga-asi.com, Andrew.Snedeker@ga-asi.com, Anastacia.Macallister@ga-asi.com, Rey.Nicolas@ga-asi.com

ABSTRACT

For the Department of Defense (DoD) to actualize its $3.3 billion in ongoing investments in artificial intelligence (AI) and machine learning (ML), it is crucial to establish a quality assurance (QA) cycle that meets the high demands of this rapidly evolving field. Current QA processes rely heavily on manual input, making them costly and prone to human error. Continuous integration and continuous delivery (CI/CD) tools, which are commonly used in software development, offer a potential solution to automate QA tasks and accelerate deployment cycles. However, adapting CI/CD processes for AI/ML introduces unique challenges, such as bridging the gap between simulation environments and real-world deployment. To fully leverage CI/CD for AI/ML warfighting solutions, additional critical components, like modeling and simulation (M&S), must be considered when architecting the solution. Effectively overcoming this QA problem will be vital for the DoD to develop AI systems that are scalable, reliable, and capable of performing at the highest levels in mission-critical scenarios.

This paper details how to architect and implement a CI/CD solution following the AI software development life cycle (SDLC), with a focus on M&S to enhance testing coverage for AI software. We address current reliance on manpower over automation and show through a cognitive modeling technique, GOMS (Goals, Operators, Methods, and Selection rules), that this method enhances testing coverage while decreasing human computer interactions by up to 80 to 90%. Ultimately, results show an exponential decrease in human in the loop testing time, all while enhancing testing coverage using automated M&S testing processes. This compelling evidence demonstrates to the DoD the need for well architected and extensive CI/CD processes to support warfighters in rapidly changing environments, who rely on AI/ML systems.

ABOUT THE AUTHORS

Victoria Dorn is a Machine Learning Engineer at General Atomics Aeronautical Systems working on Autonomy and AI solutions. Her work focuses on creating the tooling and infrastructure for productizing machine learning algorithms. Throughout her career, Victoria has made contributions in various areas such as automated labeling, synthetic data generation, computer vision algorithms and prognostic health management systems. She received her B.S. in Computer Science and Engineering from Lehigh University and is pursuing a M.S. in Applied Artificial Intelligence from University of San Diego.

Andres Ulloa is a Machine Learning Engineer for General Atomics Aeronautical Systems working on supporting multiple programs across the company's technology portfolio. Throughout his career Andres focused on bringing products from prototypes to production in fields such as robotics, computer vision, and autonomy. Andres holds a bachelor's degree from the Rochester Institute of Technology (RIT) in Biomedical Engineering.

Andrew Snedeker is a Software Developer at General Atomics Aeronautical Systems (GA-ASI). He primarily works using C++ on Autonomy and AI Solutions. Before working at GA-ASI, Andrew worked at Raytheon in Tucson, Arizona. At Raytheon, Andrew worked as a Systems Engineer doing Air-to-Air Kill Chain Simulation Software Development. Andrew holds a bachelor's degree in Computer Science from New Jersey Institute of Technology.

Anastacia MacAllister, Ph.D., is Technical Director of Autonomy and Artificial Intelligence for General Atomics

Aeronautical Systems (GA-ASI). Her work focuses on prototyping and developing novel machine learning algorithms using sparse, heterogenous, or imbalanced data sets, and exploratory data analytics. She has provided key contributions in prognostic health management, human performance augmentation, advanced sensing, and artificial intelligence for future warfare strategies. Dr. MacAllister has published over two dozen peer reviewed technical papers, conference proceedings, and journal. Dr. MacAllister holds a B.S from Iowa State University (ISU) in Mechanical Engineering, and an M.S. and Ph.D. from ISU in Mechanical Engineering and Human-Computer Interaction.

Rey Nicolas is Director of Software, Autonomy and AI Solutions and leads GA-ASI's Autonomy and AI teams developing next generation Autonomous Command and Control (C2) systems, Autonomous Processing, Exploitation, Dissemination (PED) systems, and AI edge processing for flight and sensor autonomy, air-to-air combat, and air-to-ground Intelligence, Surveillance, and Reconnaissance (ISR) missions. Rey is also an Executive Committee Leader for SAE Standard Works that is developing AI in Aviation Safety Standards. Previously, Rey worked for Intel Corporation and Motorola/Google leading AI R&D and product development for multiple product lines in the smart and connected home, autonomous driving, and healthcare applications. Rey holds a bachelor's degree in Mechanical Engineering from University of California San Diego, Master's in Computer Science from the University of Illinois, and MBA from San Diego State University

Optimizing Defense AI with Simulation-Driven CI/CD

Victoria Dorn, Andres Ulloa, Andrew Snedeker, Anastacia MacAllister, Rey Nicolas
General Atomics
Poway, CA
Victoria.Dorn@ga-asi.com, Andres.Ulloa@ga-asi.com, Andrew.Snedeker@ga-asi.com, Anastacia.Macallister@ga-asi.com, Rey.Nicolas@ga-asi.com

INTRODUCTION

As artificial intelligence (AI) and machine learning (ML) become integral to modern warfighting systems, the Department of Defense (DoD) faces a critical need to operationalize these technologies at scale. Despite over $3.3 billion invested in AI/ML development, the deployment of these systems remains constrained by fragmented manual quality assurance (QA) processes that are incompatible with the iterative and data-driven nature of machine learning. These legacy workflows introduce latency, increase operational risk, and lack the agility needed for continuous delivery of mission-relevant AI, leading to reduced warfighter trust and degraded mission performance. To address these limitations, this paper proposes a scalable CI/CD architecture for AI/ML in defense, emphasizing modular integration, modeling and simulation (M&S) testing, and continuous validation to deliver reliable performance under operationally relevant scenarios.

In conventional software engineering, CI/CD pipelines are foundational to QA, enabling automated testing, integration, and delivery. However, applying these pipelines directly to AI/ML workflows falls short in addressing AI's unique QA challenges. Unlike traditional software, AI systems require validation not only of code, but of behavior across dynamic and uncertain environments. The AI development lifecycle encompasses a broader range of components, from data pipelines and model training to deployment infrastructures, none of which are easily validated through conventional test cases. Additionally, in defense settings, real-world testing is often constrained due to cost, security restrictions, or ethical concerns, making comprehensive validation even more difficult. Conventional CI/CD pipelines alone cannot address these gaps. To bridge this divide, we introduce a modular, plug-and-play framework that integrates with CI/CD pipelines and leverages high-fidelity simulations to approximate real-world conditions. This approach facilitates adaptive, mission-driven testing that keeps pace with evolving operational requirements.

A field of practice, MLOps, has emerged to bridge the gaps in deploying machine learning by adapting DevOps principles to ML workflows, offering automation for model deployment and monitoring. For defense-specific needs MLOps still lacks the deep integration with simulation-in-the-loop validation, which we have identified as essential for testing AI systems in complex, high-stakes scenarios. While some progress has been made in specific domains such as reinforcement learning (RL), where simulation is inherently part of the training loop, broader and more robust integration across the ML lifecycle is still lacking. As a result, the ability to build, validate, and certify AI systems that are both high-performing and reliably auditable remains limited, posing important concerns for DoD stakeholders.

While we have only briefly outlined some of the challenges, this paper delves deeper and discusses our proposed AI-centric CI/CD architecture that automates the AI SDLC. Recognizing the limitations of field testing in defense settings, our solution prioritizes simulation-driven testing and validation. It is architected for rapid adaptation, allowing for the seamless incorporation of evolving mission scenarios and high-fidelity simulation environments. To measure the reduction in the complexity of human-computer interactions between the previous manual QA process and our proposed solution, we introduce a cognitive modeling method called GOMS (Goals, Operators, Methods, and Selection rules). This provides a quantitative basis for evaluating the efficiency gains and QA improvements enabled by automation. The result is a scalable foundation for deploying AI capabilities in mission-critical environments that support the DoD's objective to transition from research-driven prototypes to operationally integrated, continuously improving AI systems.

BACKGROUND

Quality assurance (QA) for AI/ML defense systems demands far more than verifying model accuracy or successful deployment. These systems must perform reliably under adversarial conditions, degraded inputs, and mission-specific constraints that traditional QA workflows are not designed to handle. Moreover, the need for accountability, traceability, and real-time performance in complex environments call for a system-level approach to model development and validation. This background section outlines three foundational areas that help inform our architecture: (1) the limitations of traditional QA workflows, (2) the role and current shortcomings of MLOps CI/CD integration, and (3) the critical need for simulation-in-the-loop testing to evaluate AI behavior in realistic operational scenarios.

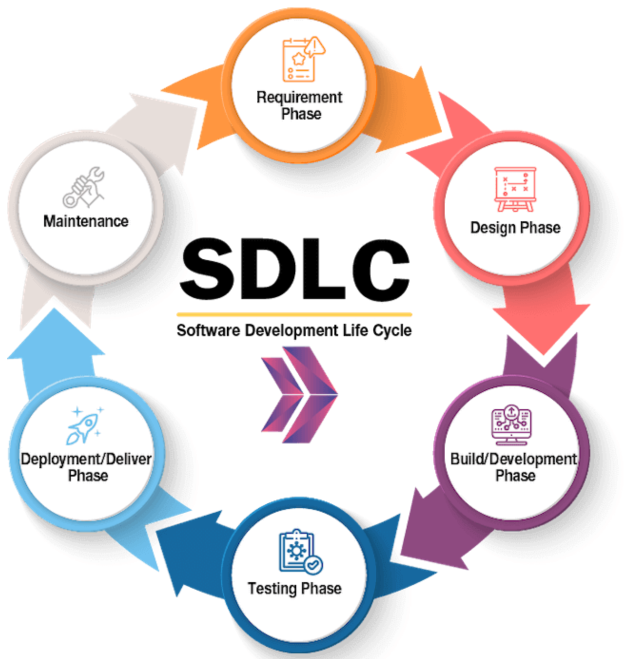

Figure 1. Waterfall SDLC model: A linear and structure approach to software development, where each phase flows into the next: Requirements, Design, Implementation, Testing, Deployment, and Maintenance.

Evaluation of Traditional QA Workflows

Traditional software development life cycles (SDLC), such as Waterfall and Agile, have long provided structured approaches to software engineering and quality assurance (QA). Waterfall methods emphasize sequential development, with QA typically introduced in the later stages, whereas Agile promotes iterative development with QA integrated continuously throughout the process. In both approaches, QA relies on deterministic testing, where functionality is validated against predefined input-output relationships using static test suites. While effective for conventional enterprise application, this method falls short for AI/ML systems, where the vast space of possible inputs makes it impractical to exhaustively define all test cases.

The transition from rule-based to data-driven systems introduces complexity at every layer of the software stack. ML models are not explicitly programmed but learned from data, making behavior less predictable and more sensitive to the underlying distributions. Issues such as data drift, class imbalance, overfitting, and adversarial vulnerabilities are difficult to detect using conventional QA techniques. Moreover, the AI development lifecycle is far more dynamic, encompassing continuous data acquisition, feature engineering, model experimentation, training, tuning, deployment, and monitoring. These stages are often non-linear and require tight feedback loops, making traditional, stage-gated QA pipelines slow, brittle, and misaligned with the needs of modern ML development (Côté et al., 2024).

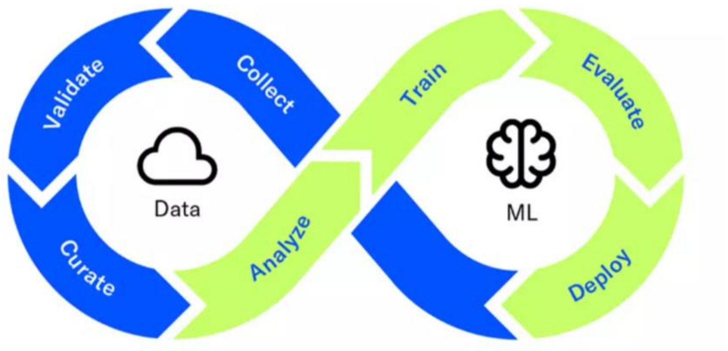

Figure 2. MLOps Lifecycle: An end-to-end view of modern machine learning pipelines, illustrating the iterative flow of a model lifecycle.

In response to these challenges, MLOps surfaced as a formal discipline, bringing DevOps principles to machine learning. MLOps frameworks introduce tools and best practices for automating and orchestrating ML workflows. As illustrated in Figure 2, MLOps encompasses key stages such as model training, versioning, deployment, and performance monitoring. This structured approach has enabled a level of reproducibility, scalability, and operational stability that was previously difficult to achieve in ML pipelines (Chatterjee et al., 2022). Core MLOps practices such as experiment tracking, model registries, and automated retraining pipelines have addressed critical pain points, particularly in commercial settings where rapid iteration and minimal downtime are essential (Tatineni & Rodwal, 2022). While still evolving, MLOps provides a strong foundation for building reliable and production-ready ML systems.

However, we recognize that the traditional ML workflow, which has been widely adopted since the early 2010s, offers only a baseline framework and falls short in addressing the unique quality assurance (QA) challenges posed by high-stakes domains such as defense. Although MLOps pipelines significantly streamline deployment and monitoring, they lack built-in capabilities for rigorous stress testing under adversarial conditions, degraded inputs, or dynamic mission parameters (Chowdary et al., 2024). In contrast, reinforcement learning (RL) frameworks inherently incorporate simulation environments and continuous feedback loops during training, offering a more adaptable QA structure. Yet even RL pipelines tend to prioritize performance optimization over formal assurance, and their QA practices are seldom generalized across broader ML workflows.

Another key limitation of traditional MLOps workflows is their lack of support for modular high-fidelity simulation environments, which are essential in defense where real-world testing may be infeasible due to cost, security, or ethical constraints. As Johnson (2024) notes, robust QA in such contexts requires simulation-in-the-loop testing that is context-aware, domain-specific, and continuously refined to reflect operational realities. Moreover, end-to-end traceability is rarely enforced in commercial MLOps pipelines, yet it is essential for certifying mission-critical systems and ensuring accountability in operations where human lives or strategic outcomes are at stake. As Avgeriou et al. (2023) emphasize, the absence of traceability compounds technical debt in ML systems, making long-term QA and system evolution increasingly difficult.

Thus, while MLOps has significantly improved automation, reproducibility, and pipeline reliability, it still lacks integrated mechanisms for comprehensive QA tailored to AI/ML's unique risks and uncertainties. In defense, where failure can have catastrophic consequences, these gaps underscore the urgent need for a more advanced QA paradigm, one that embeds simulation-in-the-loop validation, enables modular testing across system components, and enforces rigorous governance over the full AI lifecycle (Chatterjee et al., 2022; Chowdary et al., 2024).

CI/CD for AI/ML Systems

Continuous Integration and Continuous Delivery (CI/CD) practices have become foundational in modern software development, enabling teams to build, test, and release code faster and more reliably. However, adapting CI/CD to AI and machine learning (ML) systems introduces a new set of engineering challenges that traditional software delivery pipelines were never designed to handle. AI/ML systems are inherently more dynamic due to their dependency on data, as discussed prior. Unlike deterministic software applications, their non-static components (i.e. datasets, feature pipelines, models, and deployment conditions) can evolve independently. Additionally, DoD organizations are increasingly seeking to reuse AI/ML systems across multiple operational domains, each with unique constraints and risks. These variations introduce new quality assurance (QA) demands that cannot be met through static testing alone.

To address this, CI/CD pipelines must evolve to include simulation-driven testing. Simulations provide a controlled yet realistic environment to evaluate how AI systems perform under diverse, often unpredictable, conditions. When tightly integrated into CI/CD workflows, simulation-in-the-loop testing enables early identification of failure modes, supports domain-specific risk assessment, and increases trust in model behavior before real-world deployment. For warfighter applications, this level of pre-deployment assurance is critical since performance failures in the field are not just technical issues; they can have life-threatening consequences.

In parallel, ensuring the reproducibility of AI/ML models remains a persistent challenge. Small changes in training data, hyperparameters, or system infrastructure can lead to inconsistencies that undermine trust and accountability. As a result, CI/CD for AI must extend beyond code-level testing. Pipelines need to incorporate continuous data validation, automatic retraining triggers, version-controlled model artifacts, and behavior monitoring during and after deployment.

While our current system addresses some foundational needs, key challenges remain, particularly in validating and managing AI-specific risks during deployment. Instead of attempting to solve the entire DevOps problem for AI, our proposed contribution focuses on a specific area: the design of a modular and simulation-capable CI/CD framework tailored for high-stakes AI applications. This includes targeted support for interface alignment, interoperability, and simulation-backed validation.

Simulation in AI/ML Testing

Modeling and simulation play a critical role in testing AI/ML systems by enabling controlled, repeatable environments to evaluate system performance across a variety of operational conditions. High-fidelity simulations aim to replicate real-world dynamics, making it possible to test complex behaviors and interactions that may be costly, unsafe, or impractical to evaluate in physical environments. This is especially vital in mission-critical domains. Simulators have been adopted to validate perception, planning, and control systems under realistic environmental constraints, significantly reducing both cost and time to test (Shah et al., 2017).

Despite their benefits, however, simulations are not a perfect substitute for real-world testing. A key challenge is the "sim-to-real gap," where discrepancies between the simulated environment and real-world conditions can lead to degraded system performance upon deployment. These discrepancies may arise from model inaccuracies, unmodeled real-world variability, or oversimplified assumptions, resulting in QA processes that fail to capture critical edge cases. Moreover, high-fidelity simulation itself is computationally expensive and requires ongoing validation and calibration to ensure test reliability (Fabiani et al., 2023).

To mitigate these risks, our approach positions simulation as a critical component of a broader AI/ML QA strategy, not as the sole focus. Rather than investing exclusively in the development of new high-fidelity simulations, we prioritize the framework to include rigorous verification and validation of simulation models, the continuous generation of diverse scenarios to test edge cases, and the incorporation of human-in-the-loop feedback to uncover blind spots in system behavior. A key design principle of our QA architecture is modularity. This allows it to integrate with a wide range of simulation environments depending on the application domain, system requirements, and available infrastructure. By combining these capabilities with CI/CD workflows, we enable scalable, automated testing that better aligns AI performance with real-world operational needs.

AI-CENTRIC QA ARCHITECTURE DESIGN

Our architecture is built around reproducible components that integrate seamlessly into an end-to-end DevSecOps environment. The system enables AI software components to be built, tested, and deployed against mission scenarios while preserving traceability, observability, and support for rapid iteration. A defining feature of this architecture is its two-part design: the first part establishes a traditional CI/CD pipeline for component-level validation, while the second part integrates scenario-driven simulation into the broader continuous integration and deployment workflow.

Part One: Traditional CI/CD for Component Validation

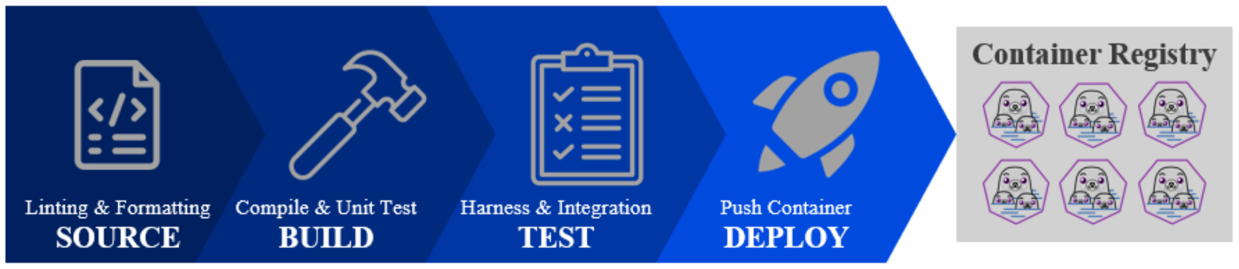

Figure 3. Continuous integration and continuous deployment pipelines with source, build, testing and deployment tools are important to a robust CI/CD testing infrastructure.

The first part of the pipeline architecture focuses on implementing standard CI/CD practices, incorporating source control, automated build processes, modular testing, and deployment automation. As illustrated in Figure 3, the pipeline provides a structured and repeatable path from development to deployment, serving as the foundation for reliable AI system delivery. Each AI system component undergoes independent validation within this CI/CD framework prior to any integration with higher-level simulations. The process begins with source control, leveraging tools such as Git to version, track, and review changes to code configurations. This ensures traceability and enables collaborative development workflows. In addition, we enforce custom rule sets aligned with DoD specific standard and project-specific standards.

Once changes are committed, the build pipeline compiles and packages the components into containerized artifacts, making them portable, isolated, and reproducible across different runtime environments. This containerization facilitates environmental consistency, a critical requirement for downstream simulation validation. Following the build stage, the pipeline conducts automated unit and integration tests. These tests validate the functionality, correctness, and specification compliance of each component in isolation. This early-stage testing is designed to catch errors before models enter complex integrated environments. This environment also serves as the point where Future Airborne Capability Environment (FACE) alignment is measured, ensuring components conform to standard interfaces and Model-Based Systems Engineering (MBSE) models. If components pass all quality gates, they are automatically deployed to a staging environment that mimics production-like conditions.

Part Two: Simulation-Driven Integration Testing

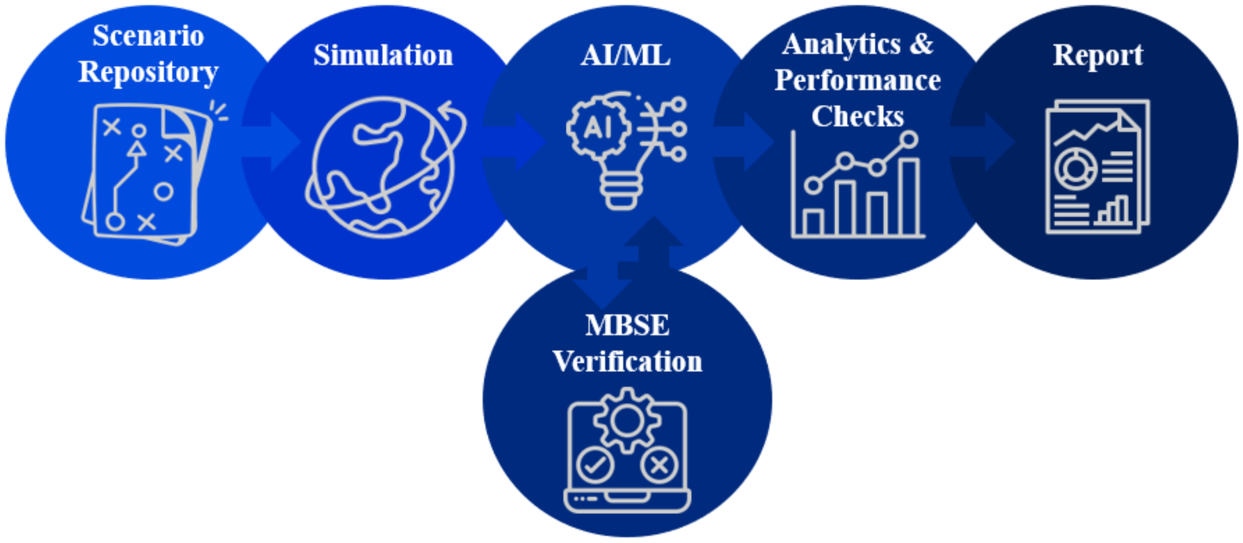

Figure 4. Simulation-driven integration testing architecture that goes beyond building and deploying to proactively detect failures and system-level integrity.

The second part of our architecture enables automated, simulation-based tests that serve as intelligent smoke tests. These tests validate the full integration of components, interfaces, and behaviors defined in MBSE models, executed within comprehensive scenario-based setups. By leveraging high-fidelity simulations that emulate mission-relevant conditions, our system assesses behavior, performance, and reliability. This provides stakeholders with detailed, actionable reports for each release candidate.

The process begins with a structured mission scenario that defines key parameters such as terrain, adversarial presence, sensor configurations, environmental dynamics, and operational constraints. Once the scenario is defined, the pipeline provisions a high-fidelity simulation environment using Infrastructure-as-Code (IaC) principles. This approach ensures reproducibility and consistency across test runs while maintaining full transparency into the test configuration.

The simulation environment acts as a testbed for system-level validation, enabling AI components to operate in complex, realistic settings. It supports the evaluation of emergent behavior, performance under stress, rare edge cases, and interactions with other system elements. The primary goal at this stage is not only to confirm functional correctness, but also to evaluate behavioral trustworthiness and operational readiness. Simulation testing can reveal extraneous actions, unintended behaviors, or failures to comply with mission constraints that may not surface in more limited testing environments.

Throughout the simulation, telemetry, interface logs, and performance metrics are collected to provide detailed visibility into how AI components interact within the operational pipeline. This data is critical for verifying interface alignment and interoperability under realistic, mission-representative conditions. After execution, the system produces a set of targeted test results that have the potential to identify integration anomalies, behavioral drift, and protocol violations. These results play a central role in QA by helping ensure that an AI release candidate would behave predictably, integrate effectively with other systems or human teammates, and do not introduce high levels of hidden risks into mission-critical environments. For the warfighter, this level of assurance supports operational trust, system reliability, and the safe use of AI in uncertain and dynamic conditions.

Together, these elements form the foundation of a CI/CD framework designed to support dependable AI integration in operational environments. By emphasizing interface alignment, interoperability, and simulation-based validation, the framework addresses a key gap in existing DevOps tools, which often fail to capture system-level behavior and operational risk. The architecture enables incremental testing and automated simulation in the loop testing, which provides a structured scalable approach to AI deployment while maintaining confidence in system performance. In the next section, we present results from applying this framework in a representative simulation scenario to evaluate its impact on improving efficiency and reducing development costs.

RESULTS AND DISCUSSION

This section evaluates the effectiveness of our proposed AI-centric QA pipeline by comparing it with a traditional manual workflow for testing and deploying a rudimentary AI behavior. The focus is on demonstrating how automation reduces overhead, minimizes human error, and accelerates delivery timelines in operationally relevant mission contexts.

Experiment Setup and Methodology

To evaluate the impact and efficacy of our AI-centric CI/CD pipeline, we conducted a comparative study between two workflows: a manual testing and deployment process (used as the baseline) and our redesigned automated pipeline. The objective was to measure the operational and cognitive efficiency of each workflow using a controlled setup around a representative AI capability. The AI behavior selected for experimentation was designed to output basic flight commands without any higher-level dynamic flight planning.

The manual process involved several human-driven steps. Previously software developers were responsible for configuring the test environment, loading the model and data, selecting test parameters, executing scripts, interpreting logs, archiving results, and issuing deployment builds. Each of these steps introduced variability based on user expertise and attention to detail, often requiring repeated interactions with command-line tools, configuration files, and test logs.

In contrast, the automated CI/CD workflow is designed to replace these manual steps with a standardized pipeline triggered by a code push update. Once triggered, the pipeline handled schema validation, simulation performance, and compliant deployment to staging or production environments. Logs and artifacts are now generated, stored, and tagged with version identifiers for traceability. By design, this workflow minimized human interaction, reduced time to deployment, and improved reproducibility and observability across iterations.

GOMS Comparison

To quantitatively assess the difference in efficiency between the manual and automated workflows, we applied the GOMS model for human-computer interactions. GOMS breaks down a task into primitive operations such as keystrokes, mouse movements, mental decisions, and system waits. Each operation is assigned an average execution time, enabling us to estimate the total cognitive and physical workload required to complete a task. This GOMS-based task analysis demonstrates a reduction in execution time and a dramatic drop in cognitive load and variability. For teams managing multiple models or deploying updates frequently, this time saving compounds quickly, reducing the burden on engineers and enabling faster iteration without sacrificing quality.

Summary of GOMS Analysis for the Manual and Automated Process					
	Manual AI Release		Automated AI Release		Time Reduction With Automation
	Method	GOMs Time	Method	GOMs Time	
Build and Packaging	Develop Code Configure parameters Trigger build Monitor Logs Archive Artifacts	2-3 hours /per component	Develop Code Auto-Trigger via CI	<5 min (automated)	95-98% reduction
Unit Testing	Open test environment Select test suite Execute test Review Test Outputs Report Issues Redevelop if Necessary	4-9 hours (1-2 hrs user time) /per cycle, manual validation	Auto-Trigger via CI Review results Redevelop if necessary	<5 min (automated)	90-99% reduction (for user)
Interface Testing	Manually developing test case Lauch interface tests Parse logs Compare outputs Document deviations Release report	2-3 days per component	Auto-trigger via CI Dashboard results of pass/fail	15-30 min (automated)	~95-99% reduction
Simulation/Integration Setup	Manually configure scenario Set up containers Check dependencies Confirm simulation performance	1-3 days per test config	Choose scenario template Auto-trigger via CI	<1 hours (parameterized scenario and orchestrated container setup)	~95% reduction
Simulation/Integration Testing	Run Test Monitor live logs Create report	1-2 hours	Auto-trigger via IaC System Logs results User reviews summary	~20-30 min (IaC-based launch)	~70-80% reduction
Total Deployment QA Time /per Iteration	Most steps done manually with actual user interaction time	10-15 hrs of user time	Mostly automated through click-and-review workflows	~2-3 hours (compute time)	~80-90% reduction

Table 1. Summary of manual versus automated actions, highlighting the significant reduction in user interaction time across development, testing, and simulation. Automation achieves approximately 80-90% reduction in manual effort, enabling faster and more reliable AI development and deployment workflows.

Analysis

The transition from manual QA and deployment workflows to automated pipelines reveals a significant shift not just in time efficiency, but also in operational consistency and human cognitive load. In the traditional workflow, building an application involves multiple discrete steps, such as writing code, compiling, and then preparing the runtime environment manually. This process, while familiar to developers, is prone to inconsistencies due to environmental differences, dependency mismatches, or human error in execution. Additionally, every build must be manually reproduced for testing or deployment environments, making it both time-intensive and error-prone.

By contrast, containerizing the build and deployment process introduces a high degree of repeatability, scalability, and speed. Once a container is defined, building

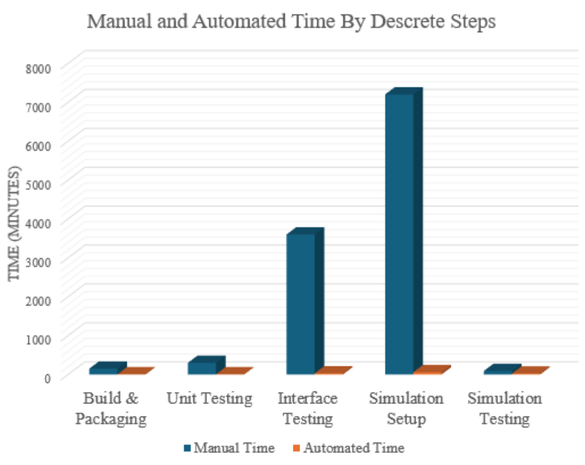

Figure 5. Comparison of manual and automated processing times across discrete steps.

and running the containerized application becomes a one-command operation. This encapsulates not only the compiled application but also its environment, dependencies, and configuration, ensuring that the application behaves the same regardless of where it is deployed. The combination of these steps with integration into CI/CD pipelines reduces human intervention to near zero after setting up. The GOMS-based analysis of this workflow indicates an X% reduction in active user time per deployment iteration, significantly freeing up engineering resources and reducing turnaround times.

Overall, the automated workflows using modern build tools align with DevOps best practices, improving maintainability and reliability across the software development lifecycle. The upfront investment in automation pays off in the form of fewer bugs due to environmental drift, more reliable testing outcomes, and drastically shortened release cycles. These improvements begin to address the gaps highlighted in the background, where traditional DevOps pipelines often fall short in handling AI-specific challenges such as interface alignment, interoperability, and risk-aware validation. By integrating simulation-backed testing in the CI/CD process, our approach provides a more structured and transparent way to manage AI deployments, helping to close the trust and safety gaps in existing frameworks.

CONCLUSION

The rapid adoption of artificial intelligence (AI) and machine learning (ML) in defense systems necessitates a shift in how quality assurance (QA) and deployment pipelines are structured. The Department of Defense (DoD) is making substantial investments to integrate AI/ML technologies into mission-critical systems, but to realize the full potential of these capabilities, current manual testing and deployment workflows must evolve. Our work demonstrates that leveraging automated continuous integration and continuous delivery (CI/CD) pipelines is a crucial step toward modernizing the software development lifecycle for AI applications.

Through the implementation of AI-centric CI/CD architecture, we have shown that automation significantly reduces the cognitive load and manual effort required to test and deploy AI models. Using the GOMS (Goals, Operators, Methods, and Selection rules) framework, we quantified a reduction of up to X% in human-computer interactions, resulting in faster deployment cycles, more consistent quality, and reduced operational risk. Furthermore, by integrating modeling and simulation (M&S) into the testing pipeline, our approach allows for better coverage of edge cases and operational scenarios that would otherwise be difficult to replicate manually.

This work has several key implications for the DoD’s AI/ML initiatives. First, it establishes that CI/CD is not just applicable but essential to the scaling and operationalization of AI capabilities in high-stakes environments. The integration of automated testing, model validation, and deployment processes ensures that AI systems are not only developed faster but also with greater confidence in their performance in real-world missions. Furthermore, the modular architecture of our solution allows for flexibility, ensuring that as AI technologies and deployment requirements evolve, the infrastructure can adapt.

However, despite these advancements, there are limitations to our approach. While we have successfully automated most of the QA and deployment pipeline, challenges remain in ensuring that the system can fully bridge the simulation-to-reality gap in mission-critical scenarios. Current simulation environments do not always capture the full complexity or unpredictability of real-world conditions. Moreover, as AI models become more complex, particularly in domains like RL or large-scale autonomous systems, new strategies will be required to account for their unique behaviors, which may differ significantly from the assumptions underlying our current pipeline.

REFERENCES

Bertolino, A. (2020). Software testing research: Achievements, challenges, dreams. *Journal of Systems and Software*, *156*, 110610. https://doi.org/10.1016/j.jss.2019.110610

Chatterjee, A., Ahmed, B. S., Hallin, E., & Engman, A. (2022). Quality assurance in MLOps setting: An industrial perspective. *arXiv*. https://arxiv.org/abs/2211.12706

Chowdary, S., et al. (2024). Towards trustworthy machine learning in production: An overview of the robustness in MLOps approach. *arXiv*. https://arxiv.org/html/2410.21346v1

Côté, P., et al. (2024). Quality issues in machine learning software systems. *arXiv*. https://arxiv.org/html/2306.15007v2

Fabiani, P., Rodriguez, A., Di Cairano, S., & Shah, M. (2023). Bridging the simulation-to-reality gap in AI testing. *IEEE Transactions on Intelligent Systems*, *38*(1), 44–55. https://doi.org/10.1016/j.eswa.2024.124310

Ferda, N. (2021, October 4). What is software development life cycle (SDLC)? *Clarusway*. https://clarusway.com/what-is-software-development-life-cycle/

Jung, S., Ha, H., Lee, J., & Yoon, S. (2022). Testing AI safety via simulation environments: A survey and case study. *Journal of AI Research and Development*, *19*(3), 221–240.

Johnson, E. (2024). Continuous testing in DevOps and MLOps: Establishing robust validation for machine learning models. *Journal of Artificial Intelligence Research*, *4*(2), 102–108. https://nucleuscorp.org/JAIR/article/view/413

Sato, Y., Teshima, S., & Ichise, R. (2023). A survey on MLOps for machine learning lifecycle management. *Journal of Big Data*, *10*(1), 1–33.

Shah, S., Dey, D., Lovett, C., & Kapoor, A. (2017). AirSim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*. https://arxiv.org/abs/1705.05065

Swanlund, S., Wilson, V., Kapoor, V., Gabbard, M., & Srinath, C. (2024). Software development life cycle. *Journal of the Society for Clinical Data Management*, *4*(1). https://doi.org/10.47912/jscdm.343

Tatineni, S., & Rodwal, A. (2022). Leveraging AI for seamless integration of DevOps and MLOps: Techniques for automated testing, continuous delivery, and model governance. *Journal of Machine Learning in Pharmaceutical Research*, *2*(2). https://pharmapub.org/index.php/jmlpr/article/view/17