WHITEPAPER

# How to Ensure Safe & Secure Software for AI/ML-Driven Embedded Systems

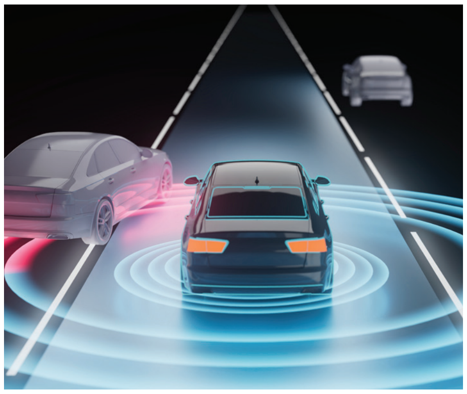
## Overview

Artificial intelligence (AI) and machine learning (ML) are reshaping embedded safety-critical systems and transforming industries like automotive, healthcare, and defense. From autonomous vehicles to medical diagnostics, AI/ML powers evolutionary technologies that enable autonomous and efficient operations. However, embedding AI/ML into embedded safety-critical systems presents unique challenges—beyond the high stakes of failure and stringent compliance requirements.

AI/ML models are difficult to implement in resource-limited systems because they demand significant processing power, memory, and real-time decision-making. Additionally, safety-critical applications require high accuracy and reliability, increasing risks due to the unpredictable behavior of AI models.

In this whitepaper, we'll dive into the challenges and advancements of AI/ML in embedded safety-critical systems. We'll also address how:

- Organizations are ensuring safety, security, reliability, and predictability of AI/ML in embedded safety-critical systems.
- Freezing models after training is critical for preventing further learning and ensure consistent behavior.
- Validation against standards like ISO 26262 and IEC 62304 is crucial for compliance.
- Software testing methods like static analysis, unit testing, and hardware-in-the-loop testing are vital to verify AI/ML reliability.
- Maintaining safety and compliance is essential for integrating AI/ML.



## Challenges of AI/ML in Embedded Safety & Security-Critical Systems

Embedded devices operate under strict energy, memory, and computational limits. Some examples include medical devices, autonomous vehicles, and industrial controls. These aren't just software projects. They're physical things with hard limits on what they can do.

In most cases, there's no room for a bulky computer chip or a fan to cool the component down. Another consideration is power. Many embedded systems run on batteries that need to last years. But AI models, especially big ones, guzzle power like a dry garden soaks up rain.

Even if you shrink the hardware and optimize the power, running complex AI nonstop can make things hot. Heat kills electronics. But it's not just about the device itself.

Embedded systems often work in harsh environments: freezing cold, scorching heat, vibrating machinery, or even underwater. If the hardware can't survive the elements, the AI becomes useless. Building AI into embedded systems isn't just about coding smarter algorithms. It's a constant tug-of-war between what the AI needs and what the physical world allows.

To reconcile AI's potential with embedded realities, developers deploy strategies to effectively reduce model size and computational overhead while preserving critical performance. Such strategies include:

» Pruning to remove less important neural pathways.

» Quantization, which compresses numerical data into low-bit formats.

We'll cover both of these strategies in more detail later in this whitepaper.

Additionally, determinism, certifiability, and resilience against adversarial conditions pose challenges that extend beyond performance optimization.

## Strategies That Balance Performance With Reliability

To address the unique demands of safety-critical embedded AI/ML, engineers deploy strategies that balance performance with reliability.

### Determinism

Safety-critical systems, such as automotive braking and flight controls, require deterministic behavior, which means consistent, predictable outputs for given inputs within guaranteed time bounds. However, AI/ML models, especially neural networks, are inherently probabilistic and often exhibit nondeterministic outputs. Consider autonomous vehicles where they require split-second obstacle detection. Any unpredictable latency could delay braking by milliseconds, violating ISO 26262 safety standards.

Freezing trained models (locking weights to prevent runtime drift) and using static memory allocation to eliminate timing variability enforces determinism. This, in turn, ensures predictable, real-time responses.

### Certifiability

Safety-critical systems must comply with stringent certification standards, like ISO 26262 for automotive or IEC 62304 for medical devices. However, AI/ML models are black boxes with opaque decision-making processes, which makes it difficult to trace decisions and prove robustness.

That being said, there's eXplainable AI (XAI). XAI refers to AI model debugging techniques using tools like LIME, SHAP, and PFI. These tools make AI models more understandable to humans by providing insights into the reasoning behind predictions and actions.

By visualizing the relationship between input features and model outputs, they enhance transparency and explainability, building trust and ensuring accountability. As a result, XAI is essential for regulatory compliance. Regulators demand evidence that outputs originate from verifiable logic rather than uninterpretable statistical patterns.

Embedded development teams can also tackle certifiability through hybrid architectures. This entails pairing neural networks with rule-based guardrails and formal verification methods to mathematically bound model behavior, enabling compliance with standards like ISO 26262 or FDA guidelines.

### Resilience Against Adversarial Conditions

Embedded systems often operate in uncontrolled environments like industrial robots and drones. They face adversarial attacks such as malicious inputs designed to fool ML models, such as perturbed sensor data causing misclassification. A specific example of this is a hacked insulin pump's ML dosing algorithm could overdose patients if adversarial inputs bypass security checks.

For resilience against adversarial conditions, there are a couple of ways to harden models.

- Adversarial training, which means exposing them to perturbed data during development.
- Input sanitization techniques, such as noise filtering, while runtime monitors track anomalies like sudden confidence drops to flag potential attacks.

Teams can further mitigate risks by securing update protocols, such as cryptographically signed OTA patches, and redundancy, like voting systems across multiple models. These measures create layered defenses that align AI/ML flexibility with the rigid safety and security requirements of embedded systems.

## Advancements in AI/ML for Embedded Systems

As mentioned above, embedded devices operate under strict energy, memory, and computational constraints. For example, a pacemaker must function for years on a single battery while running algorithms to detect arrhythmias.

Traditional deep learning models often rely on millions of parameters and clash with embedded system constraints in multiple ways.

- ✗ Their size demands excessive memory.
- ✗ Their computations drain power.
- ✗ Their complexity risks delays in real-time decision-making.

To bridge this gap, engineers use optimization techniques like the following that shrink models without sacrificing accuracy.

- ✓ Pruning eliminates noncritical neural connections.
- ✓ Quantization represents numerical values with fewer bits.

Coupled with specialized hardware like neuromorphic chips or ultra-efficient AI accelerators, these innovations are transforming embedded systems. While challenges remain, the combination of smarter algorithms and purpose-built silicon is enabling AI to thrive even in the most resource-starved environments.

### Model Compressions

Model compression is the process of transforming large, resource-intensive AI models into leaner versions that retain their core functionality. It's a critical step for deploying intelligence on embedded devices like medical sensors, drones, or spacecraft. This compression involves simplifying the model's architecture to reduce memory usage and computational demands without significantly sacrificing accuracy.

#### Pruning

A key technique is pruning, which strips away redundant or noncritical components from neural networks. For example, a model trained to recognize objects might initially have 10,000 connections between its artificial neurons. By analyzing which connections contribute least to accurate predictions, like those with near-zero weights, engineers can surgically remove 40% of them.

The result? A model that uses 60% of the original computational resources but retains 95% of its performance.

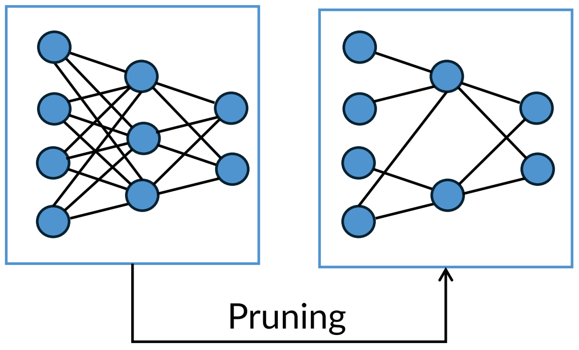
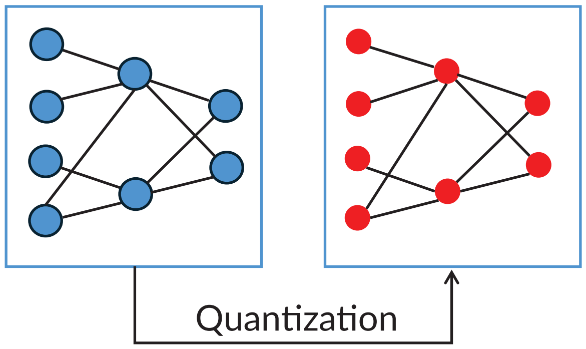


*Figure 1:*
*Neural network model pruning*

This approach isn't theoretical. NASA's Mars rovers rely on pruned models to analyze terrain in real time. During the Perseverance mission, engineers removed 40% of the neural network responsible for classifying rocks and hazards. The leaner model slashed processing time by 30%, extending daily operational windows and conserving scarce battery power—a lifesaving optimization in Mars' harsh environment, where every joule of energy matters.

*Figure 2:*
*Neural network model quantization*



## Quantization

Quantization addresses a fundamental challenge in embedded AI: the inefficiency of high-precision calculations. Neural networks typically represent parameters (weights and activations) as 32-bit floating-point numbers. It's a format precise enough for supercomputers but overkill for a smartwatch analyzing heartbeats.

Quantization simplifies this by shrinking the vocabulary of numbers the model uses. For instance, converting 32-bit values to 8-bit integers reduces memory usage by 75% and accelerates computations, as smaller numbers require fewer processing cycles. This precision trade-off is carefully calibrated, like compressing a high-resolution photo to a smaller file while retaining enough detail to recognize faces.

Wearable health trackers use quantization. Fitbit's health trackers exemplify this balance. Their sleep-monitoring AI originally used 32-bit models, which consumed enough power to drain the battery in 12 hours.

By quantizing weights and activations to 8-bit integers, engineers reduced energy consumption by 30%, a leap that enabled 24/7 heart-rate tracking and multi-day battery life. For users, this meant fewer nightly recharges and uninterrupted data collection, turning the device from a gadget into a reliable medical proxy.

Similar quantization strategies now power real-time voice assistants on earbuds and fall-detection algorithms in hearing aids, proving that smaller numbers can drive big innovations.

## Specialized Hardware

Hardware advancements, such as graphic processing units (GPUs), tensor processing units (TPUs), and neural processing units (NPUs), allow AI/ML models to run efficiently. This specialized hardware is fueling the embedded AI revolution.

General-purpose CPUs, while versatile, struggle to handle the intense computational demands of modern AI models on energy-constrained devices. This gap has spurred a wave of domain-specific architectures: chips designed not for broad computing tasks but to accelerate neural networks with surgical efficiency. Take NPUs. Unlike CPUs, they excel at parallelized matrix multiplications, which is the core math behind neural networks, and execute thousands of operations simultaneously with minimal power.

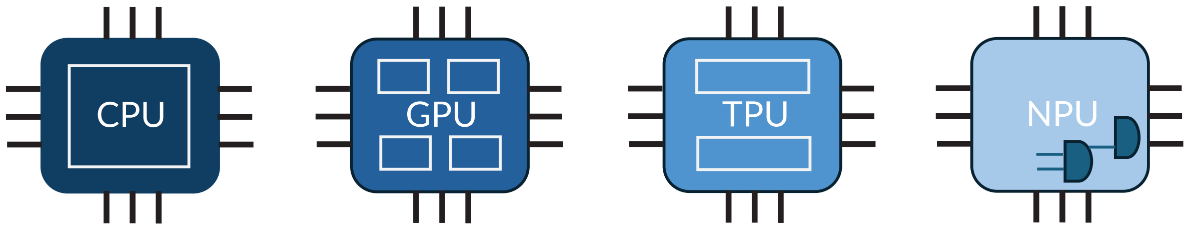


*Figure 3: Processor options for running AI/ML*

### Smartphones

Qualcomm's Snapdragon NPU illustrates how specialized hardware reshapes consumer tech. In Samsung's flagship phones, the NPU turbocharges AI-driven photography: night-mode shots that once took three seconds now process in real time, and 4K video stabilization runs continuously without melting the battery. By offloading these tasks from the CPU to the NPU, Qualcomm cut power use by 35%, turning "AI-powered camera" from a marketing buzzword into a feature users rely on daily.

## Acceptance of Risk

The probabilistic nature of AI/ML outputs, like confidence intervals, mirrors the statistical reliability metrics used for hardware components. While the sources of risk differ, the core principles of risk mitigation apply equally. Those core principles are:

- Redundancy
- Transparency
- Verification
- Adaptive safeguards

Organizations establish risk acceptance criteria for hardware failures in safety-critical systems, like defining tolerable failure rates for components like sensors or processors. Similarly, they must define risk thresholds for AI/ML-driven systems.

This process involves quantifying and qualifying the risks introduced by AI/ML in the context of the system's safety goals, regulatory requirements, and societal expectations. For example, an autonomous vehicle developer might determine that a neural network's misclassification error rate must not exceed a certain threshold to align with overall system safety targets. This is akin to how a hardware component's failure rate is bounded.

However, there are key distinctions in how teams assess and manage risks. Physical degradation, manufacturing defects, or environmental stressors are often the root cause of hardware failures. These failures are probabilistic but relatively well-characterized through historical data and standardized testing, like mean time between failures (MTBF).

AI/ML risks, by contrast, stem from algorithmic uncertainty, data dependencies, and emergent behaviors that may defy deterministic analysis. For instance, a machine learning model might perform flawlessly in testing but fail unpredictably when exposed to novel, real-world scenarios not represented in its training data.

This dynamic, nonlinear nature of AI/ML systems complicates traditional risk assessment frameworks, requiring organizations to adopt new tools, like uncertainty quantification and adversarial testing, and metrics, such as confidence scores and robustness benchmarks, tailored to algorithmic behavior.

Despite these differences, the foundational principle remains the same: organizations must define acceptable levels of risk based on the system's operational context, potential harm, and mitigation capabilities.

Regulatory standards like ISO 26262 for automotive systems and IEC 61508 for industrial safety already require rigorous risk classification for hardware and software. An example of this is Automotive Safety Integrity Levels (ASILs).

For AI/ML, analogous frameworks are emerging. ISO 21448, Safety of the intended functionality (SOTIF) for autonomous systems addresses performance limitations of AI-driven components. These frameworks emphasize that AI/ML risks must be bounded, validated, and continuously monitored to ensure they remain within acceptable thresholds—just like hardware risks.

In practice, this means organizations should:

**Align AI/ML risk criteria with system safety goals.** Define acceptable failure probabilities for AI/ML outputs, such as object detection errors in self-driving cars, which match the system's overall safety targets.

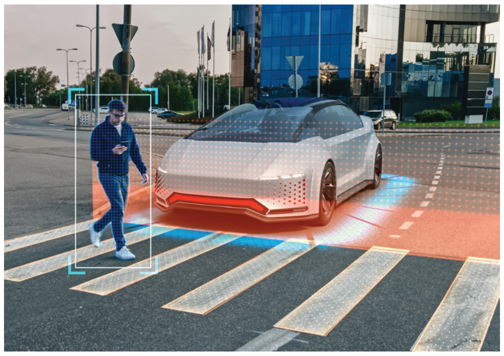
**Leverage cross-disciplinary expertise.** Combine traditional safety engineering (FMEA, fault-tree analysis) with AI-specific methods (explainability tools, robustness testing) to holistically assess risks.

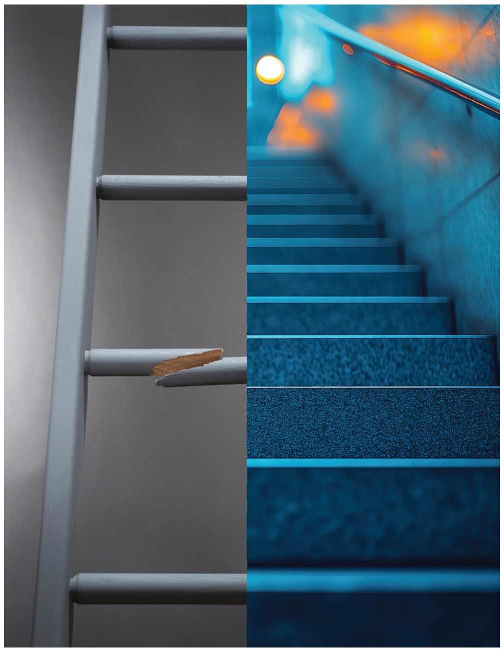
**Adopt dynamic risk management.** Unlike static hardware components, AI/ML systems may evolve via updates or retraining, for instance, necessitating ongoing risk reassessment.

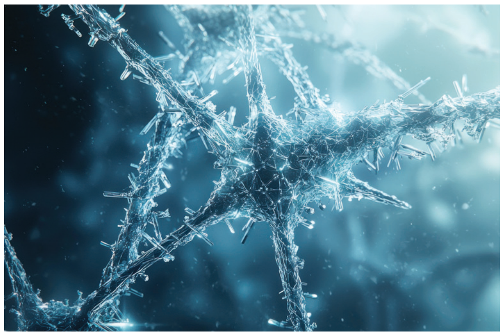
**Address ethical and societal dimensions.** AI/ML risks often include ethical concerns, like bias and transparency, that hardware failures do not. It requires broader stakeholder engagement in risk acceptance decisions.

## Addressing Safety & Compliance

Deploying AI/ML in real-time, safety-critical embedded systems, like autonomous vehicles, medical devices, and industrial robots, is challenging due to AI's inherent nondeterminism, black-box nature, and the fact that their behavior can change with different inputs or configurations.







Although the potential for unpredictability is high, with careful design, these risks can be mitigated. A common approach is to convert the model to a static and unmodifiable model after training. This is a critical step for ensuring determinism in safety-critical systems and is commonly known as "freezing the model."

Take for example automotive manufacturers designing an AI system for something really high-stakes, like a self-driving car. Lives depend on it behaving exactly the same way every single time so there are no surprises. But AI models, especially neural networks, can sometimes hallucinate and act like they've got a mind of their own. Maybe they're a tiny bit slower one day because the hardware got warm, or they round a number differently on a GPU versus an NPU. That's a nightmare for safety engineers.

### How Freezing the Model Works

This is where freezing the model saves the day. Freezing is like taking your AI model, once fully trained, and hitting Pause forever. You lock in every connection, every weight, every math operation. No more tweaks, no more updates. It becomes a fixed recipe, like a McDonald's cheeseburger, identical everywhere, whether you're in Tokyo or Texas.

Take Tesla's Autopilot, for example. Their lane-detection AI uses frozen neural networks updating them only after rigorous validation cycles. While Tesla employs over-the-air (OTA) updates for its Full Self-Driving (FSD) software, these updates undergo extensive pre-deployment testing to ensure stability. Why? Because regulators like the folks behind ISO 26262—the gold standard for automotive safety—demand proof that the system behaves exactly as tested, in every car, under all conditions.

Freezing lets them say, "We've certified this version, and it won't drift over time."

But here's the thing: freezing alone isn't enough to pass certifications.

Imagine freezing a model that uses 32-bit floating-point numbers. Even tiny differences in how chips handle decimals could cause inconsistencies, like a medical robot calculating a 0.1-millimeter incision as 0.1000001 one day and 0.0999999 the next. For certifications, that's unacceptable. That's why engineers pair freezing with quantization, shrinking those 32-bit numbers into 8-bit integers. It's like swapping a shaky ladder for a solid staircase, fewer chances to slip, more confidence in every step.

Certification bodies also hate black boxes. They want to know exactly what's running on the device. A frozen model becomes a static file. For example, a .tflite file (TensorFlow Lite model) or .onnx file (Open Neural Network Exchange model) that engineers can hand to auditors and say, "Here's the AI, line by line, byte by byte. Test it once, and it'll behave the same way in all 10 million cars."

Of course, certifications like ISO 26262 or IEC 62304 don't just care about the model. They care about the whole pipeline, data, training, and deployment. Freezing simplifies this by turning the AI into a single, unchanging artifact. Think of it like sealing a time capsule. Regulators can inspect it once, approve it, and trust it won't mutate later.

But even with freezing, you still need guardrails. A frozen AI might still make a wild guess if it sees something totally new, like a self-driving car encountering a couch in the middle of the highway. That's why systems pair frozen models with rule-based fallbacks. For example, "If the AI's confidence drops below 95%, then slam the brakes." Additionally, runtime monitors cross-check outputs against physics or common sense.

In the end, freezing is like getting a tattoo of your AI model. It's permanent, predictable, and audit-friendly. For automakers, it's not just a technical step, it's a business enabler. Without it, they couldn't legally put their cars on the road. For engineers, it's the price of admission to the world of safety-critical AI: prove your system is boringly consistent or stay in the lab.

## The Role of Traditional Verification Practices in AI-Enabled Embedded Systems

Embedded safety-critical systems that incorporate AI components must still adhere to foundational verification and validation practices, including static analysis, unit testing, code coverage, and requirements traceability. These practices remain critical to ensuring system integrity, regulatory compliance, and safety. However, their implementation may require adaptation to address the unique challenges posed by AI.

### AI-Aware Techniques

While AI introduces novel challenges, traditional verification practices remain indispensable. The solution lies in augmenting these practices with AI-aware techniques, ensuring comprehensive coverage of both code and model risks. By maintaining rigorous static analysis, unit testing, code coverage, and requirements traceability, while adapting tools and methods to address AI's unique demands, developers can achieve the robustness required for safety-critical certification and operational deployment.

## Static Analysis

Static analysis retains its importance even in AI-driven systems. AI operates within an existing architecture that includes traditional code such as control logic, sensor interfaces, and safety monitors. Teams can rigorously analyze this conventional code using established tools to detect vulnerabilities and ensure compliance with coding standards.

AI components, such as neural networks, introduce new complexities. Although the field is still in its infancy, emerging tools are beginning to address AI-specific risks, such as unsafe operators in frameworks like TensorFlow or PyTorch. Static analysis must evolve to cover both traditional software and AI framework configurations while supporting compliance with safety standards like MISRA C/C++ and CERT C/C++.

## Unit Testing

Unit testing continues to validate low-level functionality, particularly for non-AI components such as communication protocols, fault handlers, and hardware interfaces. However, AI models, which derive behavior from data rather than explicit code, demand a shift toward data-driven validation. This includes testing for edge cases, adversarial inputs, and scenario coverage.

While the AI model itself may not be amenable to traditional unit tests, the surrounding code, such as data preprocessing pipelines, inference engines, and output handlers, must still undergo rigorous unit testing to isolate and mitigate defects.

## Code Coverage

Code coverage metrics, such as modified condition/decision coverage (MC/DC), remain mandatory for certification in domains like aviation (DO-178C) and automotive (ISO 26262). These metrics ensure exhaustive testing of decision logic in conventional code.

For AI components, traditional code coverage is not directly applicable, but analogous concepts are emerging. Techniques like input space coverage and neuron activation coverage aim to quantify the sufficiency of testing for neural networks, ensuring that diverse scenarios and model behaviors are exercised during validation.

## Requirements Traceability

Requirements traceability is nonnegotiable in safety-critical systems, as it provides an auditable chain from system requirements to implementation and testing. AI complicates this process due to its data-driven, often opaque decision-making. Essential practices to maintain traceability include:

- Explainable AI (XAI)
- Model introspection
- Thorough documentation of training data, model architecture, and validation results

For example, a requirement such as “the system shall detect obstacles in low-light conditions” must map to sensor code and also to the AI model’s training data diversity and robustness testing.

### Regulatory Frameworks

Regulatory frameworks, like IEC 62304 for medical devices, implicitly mandate traceability practices, regardless of AI involvement. Hybrid approaches that combine classical verification with AI-specific methods are necessary to address both traditional software risks like memory leaks and AI-specific failure modes like dataset bias. Examples include adversarial testing, runtime monitoring, and bias detection.

Furthermore, system-level safety depends on the seamless integration of AI and non-AI components. Flaws in either can cascade into catastrophic failures.

## Summary

The integration of AI and ML into embedded safety-critical systems presents both opportunities and challenges. While AI-driven automation enhances system capabilities, it also introduces complexities such as nondeterminism, compliance hurdles, and security vulnerabilities. Ensuring reliability requires a combination of model optimization techniques, specialized hardware, and rigorous verification methodologies.

Techniques like pruning, quantization, and specialized hardware enable AI to function efficiently within embedded constraints. However, safety and compliance demand additional measures:

- Freezing models.
- Implementing rule-based safeguards.
- Applying verification practices like static analysis, unit testing, and coverage analysis.

These strategies help mitigate AI's inherent unpredictability and align with safety standards like ISO 26262 and IEC 62304. Ultimately, successfully deploying AI in embedded systems depends on balancing innovation with regulatory compliance. As AI-specific verification methods evolve, integrating traditional software safety practices with AI-aware techniques will be essential to ensuring AI-driven embedded systems remain safe, reliable, and certifiable in real-world applications.

## TAKE THE NEXT STEP

Talk to a compliance expert to learn how your team can safely, securely, and reliably integrate AI into your embedded software development.

### About Parasoft

Parasoft helps organizations continuously deliver high-quality software with its AI-powered software testing platform and automated test solutions. Supporting the embedded, enterprise, and IoT markets, Parasoft's proven technologies reduce the time, effort, and cost of delivering secure, reliable, and compliant software by integrating everything from deep code analysis and unit testing to web UI and API testing, plus service virtualization and complete code coverage, into the delivery pipeline. Bringing all this together, Parasoft's award-winning reporting and analytics dashboard provides a centralized view of quality, enabling organizations to deliver with confidence and succeed in today's most strategic ecosystems and development initiatives—security, safety-critical, Agile, DevOps, and continuous testing.