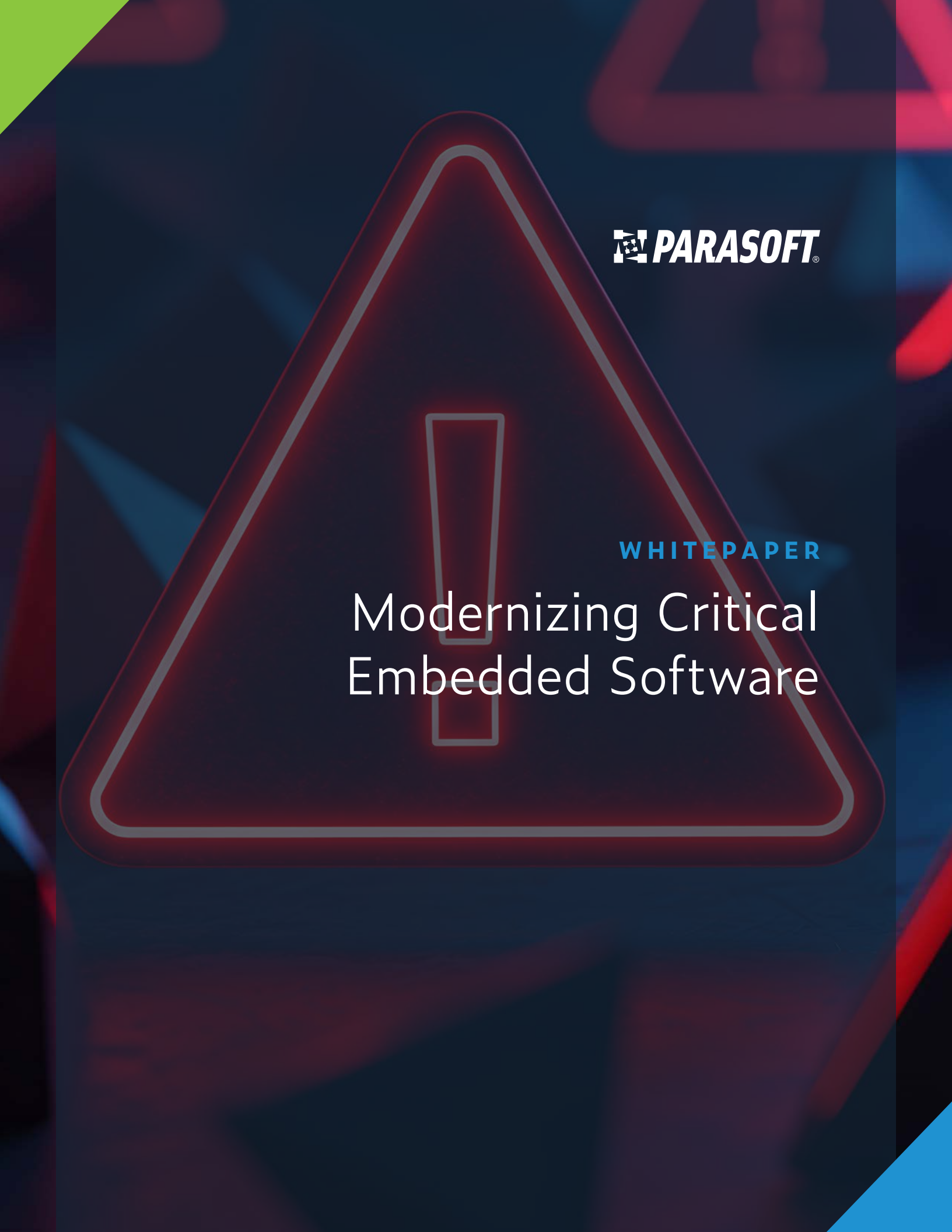
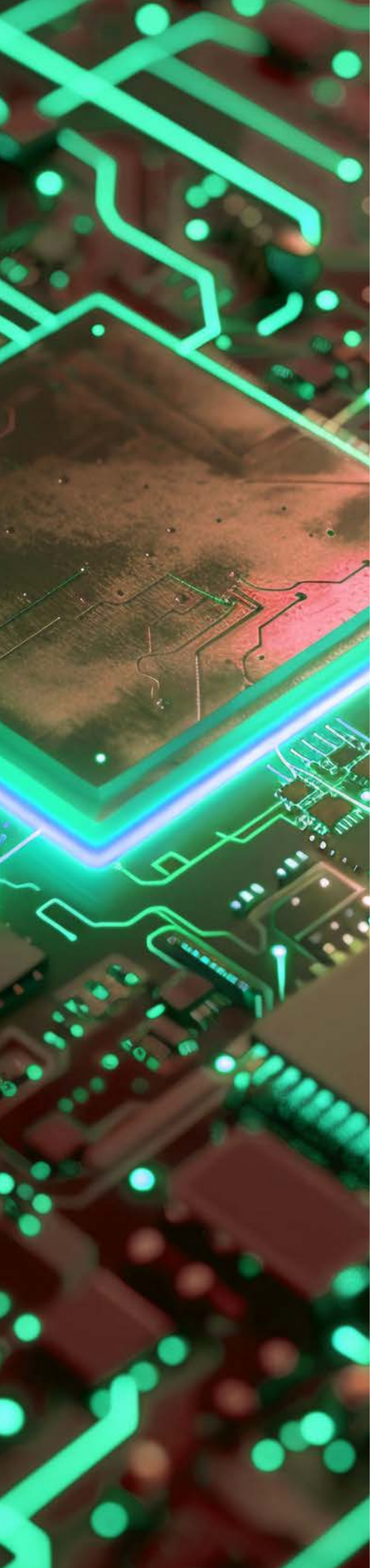


PARASOFT®

WHITEPAPER

Modernizing Critical Embedded Software



Executive Summary

Embedded software is a competitive differentiator, yet most programs still rely on legacy, hardware-bound workflows that were never designed for today's scale. This paper explains how the **enterprise world already has a head start** on the problem; why **embedded teams must modernize** now, before AI-driven code volumes escalate; and how new regulatory mandates turn compliance into schedule killers. It also shows how to do it without compromising real-time, safety-critical demands.

You'll learn about:

- **The enterprise blueprint, adapted for embedded.** Cloud-native DevOps, containerized builds, and AI assistance have already helped productivity in enterprise software. We go through which elements translate to embedded environments already, and which ones must be customized.
- **Embedded-specific hurdles.** Custom silicon, hardware-in-the-loop, real-time performance, safety standards, and C/C++ constraints that block adoption from happening.
- **The testing and compliance burden.** Why code growth, GenAI churn, and multiple standards (MISRA, ISO 26262, DO-178C, etc.) make continuous, automated validation non-negotiable.
- **A five-step modernization roadmap.**
  - Assess toolchain
  - Containerize builds
  - Automate compliance-grade testing
  - Pilot cloud CI/CD on simulation hardware
  - Scale with AI-driven prioritization
- **The hidden value of platforms.** A brief look at how having an integrated platform bridges embedded and functional domains, continuous compliance, and readies teams for DevOps speed.

Written for CIOs, CPOs, chief software officers, and functional-safety owners who are aware of the pain but don't yet have a solution, this guide delivers a concise business case and practical path forward. And it gives you the confidence that you can modernize your embedded development process without sacrificing the quality your projects demand.

Enterprise DevOps has sprinted ahead. Embedded teams are falling behind.

Cloud computing ushered in a new era of productivity for enterprise software development, catalyzing the software industrial revolution. Innovations in both deployment infrastructure and software-development methods have facilitated rapid advancements on a large scale, while simultaneously reducing nominal costs.

> ***It's time to adapt embedded software development to a modern, industrialized scale.***

Recent breakthroughs in artificial intelligence are changing the game once again, with the promise of additional, and potentially transformational, productivity boosts.[1]

Embedded systems, meanwhile, have traditionally been curated as custom solutions, intimately tailoring software to specific hardware architectures. They're also characterized by features such as real-time responsiveness and functional safety requirements not required elsewhere.

Embedded software development has not enjoyed the same pace of innovation and productivity improvements, and generally fallen short of the best practices observed in enterprise software environments.

Given the increasing importance of software as a differentiator across multiple industry sectors, **it's time to adapt embedded software development to a modern, industrialized scale.**

This white paper examines what will be needed for this transformation. It outlines the necessary adaptations to the unique characteristics of embedded systems, while also leveraging advancements seen in enterprise software—including a shift to DevOps and the integration of artificial intelligence. There's also an emphasis on the burden of testing these systems, and how embracing modern practices can help there, too.

Why now?

- Market pressure
- Regulatory updates
- Spike in AI and complexity

1 PwC: *2024 Cloud and AI Business Survey*

Several Factors Have Led to Modern Software Development

Many factors have added to the explosion of productivity in modern software. While the list below is not exhaustive, these are some of the key enablers, and provide a roadmap for the steps required to transform development of embedded systems, too.

- **Decoupling Hardware and Software on the Way to the Cloud**

  Enterprise software was once closely coupled to underlying silicon. In the 1980s and 1990s, computer systems from Sun, HP, IBM, NeXT and others operated their own proprietary operating systems on custom hardware.

  Over time, there was a consolidation towards a common silicon architecture (Intel's x86) and a common operating environment: Linux. This reduced dependencies, consolidated tools, and provided a common environment with a broad community of developers.

  Companies like VMware offered virtualization solutions that let you consolidate different systems onto a single, virtualized platform, reducing costs and improving server efficiency. Software then became further abstracted from silicon through containerization, such as Docker or Kubernetes, helping migration to cloud-native infrastructure with a focus on automation and orchestration.

  Currently, microservice architectures let developers concentrate on software features rather than infrastructure, promoting faster development cycles. They're also scalable, adaptable, resilient, and easier to maintain. Using containerized architectures only helps, enhancing these systems' portability and consistency.

- **DevOps and Build Automation**

  Cloud technology enables rapid, large-scale creation of software in environments often referred to as the “Software Factory” in the Aerospace and Defense (A&D) sector. Moving from monolithic IDEs to containerized tools, managed via scripts and build systems, has established a production line approach, synonymous with DevOps or CI/CD pipelines.

  Following cloud-native practices lets build-infrastructure leverage the cloud. This shifts code editing and compilation from developers' desktops to automated processes within the Software Factory, and taps into seemingly unlimited computing resources.

  These factories, with their standardized tools and scalable infrastructure, have greatly improved productivity. Continuous, production-ready builds now take hours or minutes instead of weeks or months. At the same time, it supports ongoing regression testing across multiple configurations and versions.

- **Development Collaboration**

  Separating hardware from software and using cloud resources has greatly helped software teams expand. Development resources are now shared across sites and borders, with containerized systems allowing independent function development.

  Integration and testing occur in a unified environment with common tools, workflows, and reporting. This scalable setup lets teams scale efficiently without the need to be co-located.

- **Development, Customized “My Way”**

  Standardizing development pipelines and tools ensures uniformity while allowing developers to work in ways that suit their preferences, boosting productivity. Modern tools, often lighter than legacy ones, can be optimized with plug-ins or custom scripts via CLIs. Open frameworks like Microsoft’s VS Code have fostered a vast ecosystem of productivity tools, enabling developers to code efficiently.

- **Open Source and Rich Frameworks**

  The trend towards increasing abstraction is not confined to the separation of software from hardware. Programming languages have also advanced to higher levels of abstraction, offering sophisticated environments with comprehensive libraries and services that enhance developer productivity.

  As these evolving software resources are shared and reused, they provide a more advanced baseline for development and innovation. Numerous languages and frameworks are available to developers as open-source platforms, fostering collaboration across projects while minimizing acquisition costs.

- **Complex Build Systems**

  Build systems, often managed by experts, are rarely considered in the development process once set up.

  However, as cloud systems grow in complexity, so do their build systems. Modern environments like Bazel efficiently track code changes and automate recompilation and relinking. This leads to faster, scalable iterative builds, reducing full build cycles and optimizing resource usage.

- **Coding and GenAI**

  Generative AI is transforming software development by automating code writing, bug detection, and improvement suggestions. This boosts productivity, letting developers focus on creative design and innovation.

  While simple tools like auto-complete are minimally intrusive to existing workflows, advanced AI methods like "Vibe Coding" can burden developers with excessive code they need to understand or reverse-engineer, which is time consuming.

When It's All Put Together

Taken together, these factors form a proven playbook: decouple software from hardware, automate everything that can be automated, and equip globally distributed teams with lightweight, customizable toolchains. Then use AI to handle what slips through the cracks. The result in the enterprise world is a development cycle measured in hours rather than months.

For leaders responsible for safety-critical, resource-constrained embedded projects, the obvious question now is how much of that playbook can we safely import. And where will the embedded-specific barriers appear? The next section unpacks those modernization challenges and pinpoints the adaptations required to capture cloud-scale speed without compromising real-time performance—or compliance.

The Barriers to Modernizing Embedded Software

Enterprise computing platforms like personal computers, servers, and cloud infrastructure are optimized for general-purpose tasks. They provide maximum computational power within cost and power limits.

Conversely, embedded systems have unique constraints, leading to specialized development processes that haven't adapted to modern methods due to some of the following barriers.

- **Custom Hardware**

  Embedded systems are deployed in a range of devices across varying applications, often custom-designed for a specific, such as a medical device.

  They use a wide range of processor architectures, as well as custom hardware interfaces for interacting with sensors or other systems around them. They require software tools tailored to these environments, such as special compilers, debuggers or other development tools.

- **Hardware-in-the-Loop**

  Even with high levels of abstraction from underlying hardware, embedded software needs to be tested on either development or deployment systems. This requires tools and workflows that allows for cross-development; development or testing on one machine, while executing and debugging code on the custom hardware platform.

- **Constrained Environments**

  Embedded systems are often deployed in environments in which there are limits on available resources, such as space, weight, power or available memory. The art of embedded development is achieving maximum and predictable performance within these constraints.

- **Real-Time Performance**

  Interacting with real-world environments often puts timing constraints on systems and their needs for responsiveness. These can include real-time behavior (a guaranteed timing response to external events), worst-case timing limits, or determinism (predictable and consistent timing behavior).

- **Functional Safety**

  Many systems operate in environments where their failure could pose a risk to their surroundings or operators. Functional safety software is created to ensure that safety-critical systems and applications work correctly, even when malfunctions occur. These systems are frequently regulated and must be developed according to industry-specific software development standards.

- **Development Languages**

  Custom hardware or system constraints often necessitate software languages optimized for hardware. Thus, C and C++ are common choices for embedded development, whereas higher-level languages may be unsuitable due to non-deterministic runtime behavior in areas such as memory management or complex I/O libraries.

The Balancing Act

Collectively, these constraints explain why embedded teams can't simply lift from the cloud playbook. Every optimization must respect custom silicon, scarce resources, hard real-time deadlines, rigorous safety standards, and the low-level languages that hold it all together.

Modernization, therefore, isn’t a matter of swapping tools—it’s a balancing act that must preserve compliance while taking advantage of the automation, scale, and collaboration gains already proven in enterprise software. The remainder of this paper shows how to strike that balance, beginning with the heaviest brake on embedded development.

The Truth About Testing—It’s Seen as a Burden

Testing is a critical step in the software development process, yet is often seen as a burden. It is a gating item for release of software products, though given the choice many developers would prefer to avoid it and focus instead on coding.

Developers often perceive testing as a distraction from the creative aspect of coding, yet it remains indispensable for ensuring the reliability and functionality of the final product.

Given the reluctance for testing, the process becomes a bottleneck in the software development lifecycle. The challenge lies in balancing thorough validation with tight deadlines, limited resources, and the increasing complexity of modern software systems.

Streamlining and automating testing procedures, and embedding them seamlessly into development workflows, is essential for alleviating this burden. These are also the steps for maintaining quality, performance and standards conformance.

Growing Software Content and Complexity

As developers adopt frameworks for baselines on advanced software, or employ higher-level languages for increased productivity, the amount of code to be tested continues to grow.

Adding GenAI to the mix compounds this growth. Research indicates that while using GenAI for code generation does not necessarily expand the code base, it appears to increase a software repository's churn as generated code is analyzed and re-factored.[2]

These contributors to software growth also highlight that developers need to review and test **code that they did not write.** This further increases software complexity and reinforces the need for testing.

The adoption of these new technologies may make testing more challenging, making it more imperative to streamline our approach to it.

Conformance to Standards

Adherence to software coding standards, such as MISRA, brings numerous benefits to the development of safety-critical and reliable systems.

These standards establish a uniform coding framework that minimizes ambiguities and enforces best practices, significantly reducing the likelihood of programming errors. They improve code maintainability and readability, making it easier for teams to collaborate, troubleshoot, and adapt to evolving requirements.

Furthermore, compliance with such standards makes system behavior more predictable, a vital aspect in safety-critical environments, and facilitates certification processes by aligning development efforts with industry regulations. By embedding these guidelines into workflows, organizations can build robust, high-quality software with reduced risks and streamlined validation efforts.

Continuous Compliance

Adhering to software functional safety standards is another foundational best practice in developing reliable and safety-critical systems. For effective and continuous conformance, teams should integrate standards compliance into their workflows from the outset, ensuring that requirements like ISO 26262, IEC 61508, or DO-178C are systematically addressed.

This involves leveraging tools for static analysis, code coverage, and requirements mapping, so teams can validate each stage of development against applicable regulations.

Collaboration between developers, testers, and quality assurance teams is essential to maintain adherence without creating unnecessary bottlenecks. Regular audits and reviews should be scheduled to verify compliance while using automated tools to streamline documentation and verification processes.

Additionally, fostering a culture of accountability and education around these standards ensures all team members understand their importance, reducing errors and simplifying certification efforts.

2 *https://arc.dev/talent-blog/impact-of-ai-on-code*

Giving Developers Access to Modern Software Development Methodologies

In the ever-evolving landscape of software development, modern methodologies such as DevOps and continuous deployment are reshaping how teams build, test, and release applications. You will do best with a comprehensive suite of tools and technologies that equip developers to meet the demands of modern development practices.

By addressing challenges in both embedded and functional software, providing robust tool integrations, and enhancing developer productivity, your team can focus on innovation and reduce risk while maintaining high standards of quality, performance and conformance to standards.

The Value of Platforms That Span Embedded and Functional

The most viable companies can provide tailored solutions that cater to the diverse needs of embedded and functional software development. In the realm of embedded systems, where safety-critical and real-time performance are paramount, it's best to use tools that ensure rigorous testing and compliance with industry standards such as MISRA, ISO 26262, and DO-178C.

These solutions help developers achieve high levels of reliability by embedding testing and validation workflows seamlessly into their development processes.

For functional software applications, you'll need advanced test automation capabilities that accelerate development cycles and improve software quality. Look for platforms that support API testing, service virtualization, and test data management—components that help your team identify defects early in the lifecycle and reduce costly rework.

By addressing the unique requirements of both embedded and enterprise-level applications, you can bridge the gap between artisanal approaches and production-scale methodologies.

Enabling the Developer-in-the-Loop

As software systems grow increasingly intricate, developers must remain actively engaged in reviewing and managing build pipelines. This involvement is crucial to ensure that insights generated by automated tools, such as static analysis, are appropriately triaged and acted upon.

With the rising volume of code content and the complexity of interdependent systems, the sheer number of warnings, errors, and recommendations produced during builds can quickly overwhelm developers.

Without their oversight, critical issues may be overlooked, potentially compromising software reliability and security. Developers bring the keen analytical skills and domain-specific understanding needed to discern which actions should be prioritized, ensuring the pipeline delivers meaningful results while maintaining efficiency.

Artificial intelligence offers a transformative way to address the burdens associated with managing increasingly complex codebases and build pipelines.

By leveraging historical data and predictive algorithms, AI can intelligently streamline the process, identifying risk-prone areas and automating the prioritization of tasks. Rather than sifting through an avalanche of redundant or inconsequential recommendations, developers can focus on actionable insights that drive measurable improvements.

AI-powered tools enhance collaboration between developers and automated systems, enabling build pipelines to scale effectively while preserving the critical human element that ensures the results are aligned with organizational goals and standards. These advances are already built into Parasoft's tool workflows, and is a continued area of focus for ongoing investment.

Where Companies Like Parasoft Fit In

Parasoft, a leader in automated testing and software quality solutions, approaches modern software development methodologies to help developers in several key ways.

- **Streamlined workflows.** By embedding testing tools at every stage of development, Parasoft removes bottlenecks and ensures that developers can focus on innovation without compromising quality.
- **Compliance simplified.** Automatic validation against industry standards and regulations reduces the manual burden on teams while improving the predictability and safety of software systems.
- **Support for continuous deployment.** Parasoft's integration into DevOps pipelines enables rapid iteration and deployment, ensuring that teams can deliver value to users without delays.
- **Enhanced collaboration.** Shared insights, reports, and dashboards foster a culture of accountability and teamwork, ensuring that quality is a shared responsibility across the organization.

Tool Integrations for Enhanced Developer Productivity

Parasoft's ecosystem of tools integrates seamlessly into modern development environments, enabling developers to leverage their existing workflows while enhancing efficiency.

Key integrations include:

- **DevOps pipelines.** Parasoft's solutions integrate into CI/CD pipelines, ensuring continuous testing and validation throughout the development lifecycle. This alignment with DevOps practices allows teams to detect and address issues early, fostering faster delivery of high-quality software.
- **Code analysis and coverage.** Parasoft integrates tools for static and dynamic code analysis into popular IDEs as well as more modular editors such as VS Code, enabling developers to identify potential errors, enforce coding standards, and achieve comprehensive test coverage without leaving their development environment.
- **Requirements traceability.** By connecting test cases to specific requirements, Parasoft ensures complete traceability, making it easier to validate functionality against specifications and comply with regulatory standards.

- **Testing collaboration tools.** Parasoft facilitates collaboration among developers, testers, and quality assurance teams by providing shared dashboards and detailed reporting. These tools streamline communication and ensure alignment across teams.
- **AI-Powered insights.** Parasoft leverages artificial intelligence to optimize testing efforts. By analyzing historical data and test results, its tools can identify risk areas, prioritize test execution, and reduce redundant efforts, boosting overall productivity.

Embedded Meets Enterprise: The Best of Both Worlds

Parasoft's ability to address both embedded and enterprise applications provides organizations with a unified approach to software quality.

By combining the meticulous attention to detail required for embedded systems with the scalability and agility of enterprise solutions, Parasoft equips teams to thrive in diverse development environments. Whether the goal is to build safety-critical systems or rapidly iterate on mission-critical enterprise applications, Parasoft's tools offer the flexibility and support needed to succeed.

Old Meets New

As software development methodologies continue to evolve, the need for adaptable, efficient, and robust tools has never been greater. Parasoft's comprehensive suite of solutions enables teams to embrace modern practices such as DevOps while addressing the complexities of embedded and functional software development.

We believe this will become increasingly important as applications and devices become ever more connected and dependent on each other.

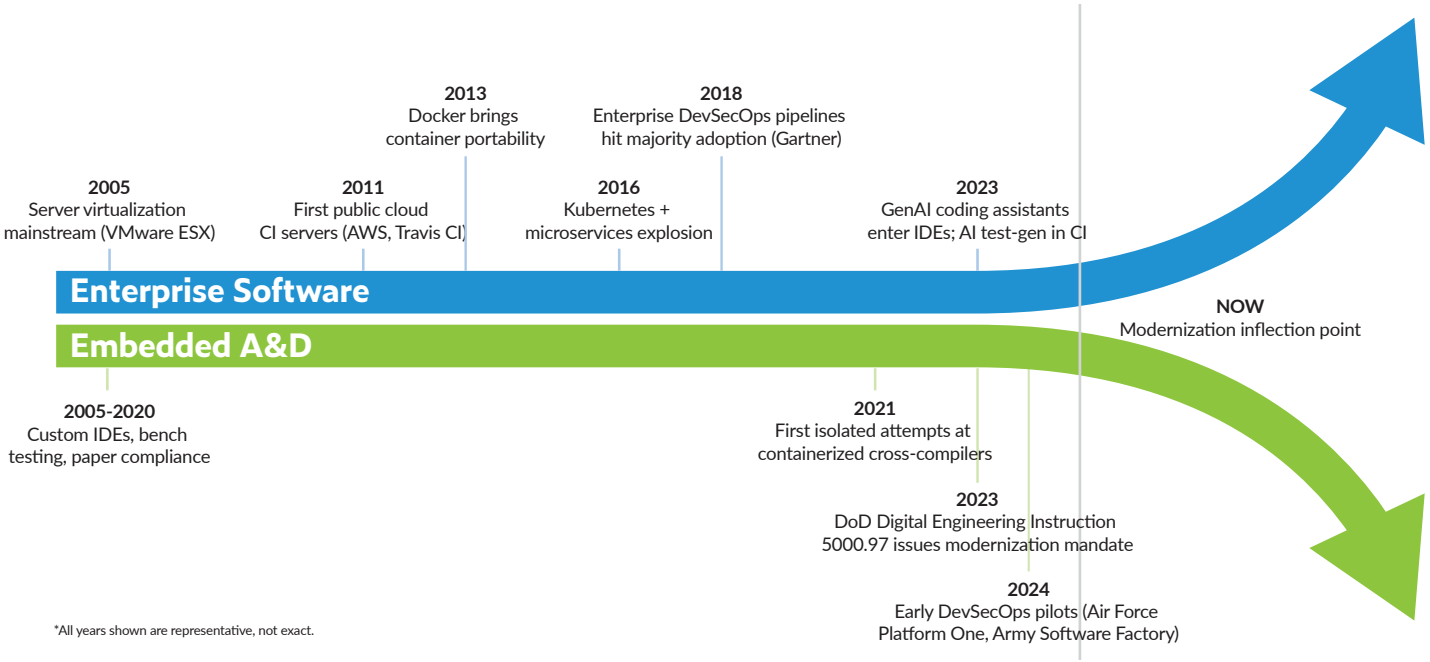
Through seamless tool integrations, AI-powered insights, and a commitment to quality, Parasoft not only enhances developer productivity but also ensures that organizations can deliver reliable and innovative software in today's fast-paced environment.

Because Parasoft has followed a developer-focused philosophy that has helped support embedded developers for over 35 years—which included traditional IDE-based workflows—it was quick to recognize the required shift to DevOps. This led to it investing and expanding its product portfolio to satisfy the needs of both approaches. This provides developers with choices over the migration from traditional to modern workflows.

With a portfolio that spans embedded and functional software, you can bridge best-practices from each domain to the other. We believe this will become increasingly important as applications and devices become ever more connected and dependent on each other.

In the case of automotive software, for example, the increasing software content in vehicles requires a shift to software defined vehicle architectures where embedded software deployed in vehicles is managed and updated by cloud-based services such as over-the-air (OTA) updates.

Examples such as these abound across every industry, and mark another evolution of the complexity of the ever-changing software landscape.



How to Move Forward

Your roadmap will look different from every other team's. See what industry-specific options you have for your embedded workflow—industries like aerospace, automotive, medical, and more—at parasoft.com/industries/embedded.

Modernizing embedded workflows is no longer optional. It's now a competitive imperative, and therefore a compliance imperative.

TAKE THE NEXT STEP

Contact us to speak directly with a solutions architect.

For any product or example of AI discussed here, our experts can provide a free, custom demo tailored to your hardware, safety standards, and workflow.

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.