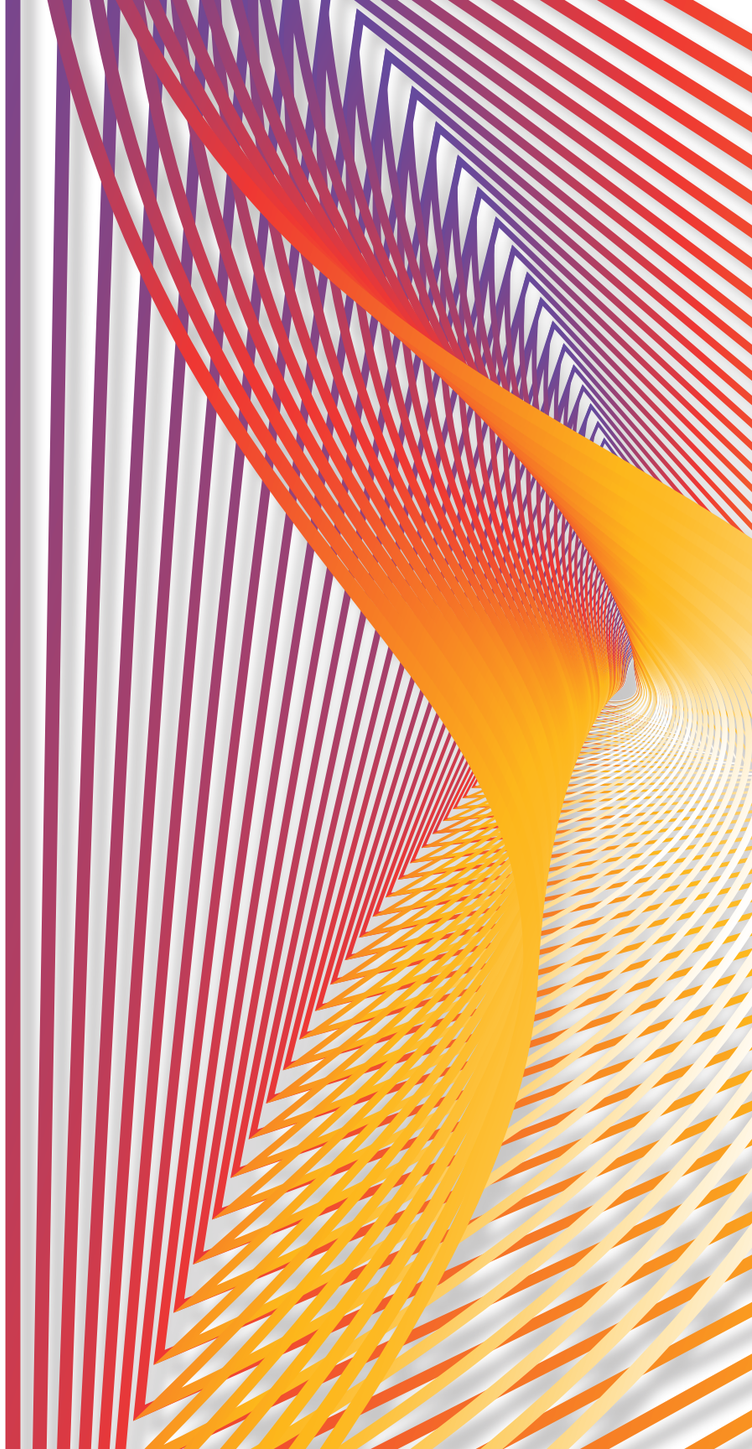
BLACK DUCK®

# THE STATE OF EMBEDDED SOFTWARE QUALITY AND SAFETY

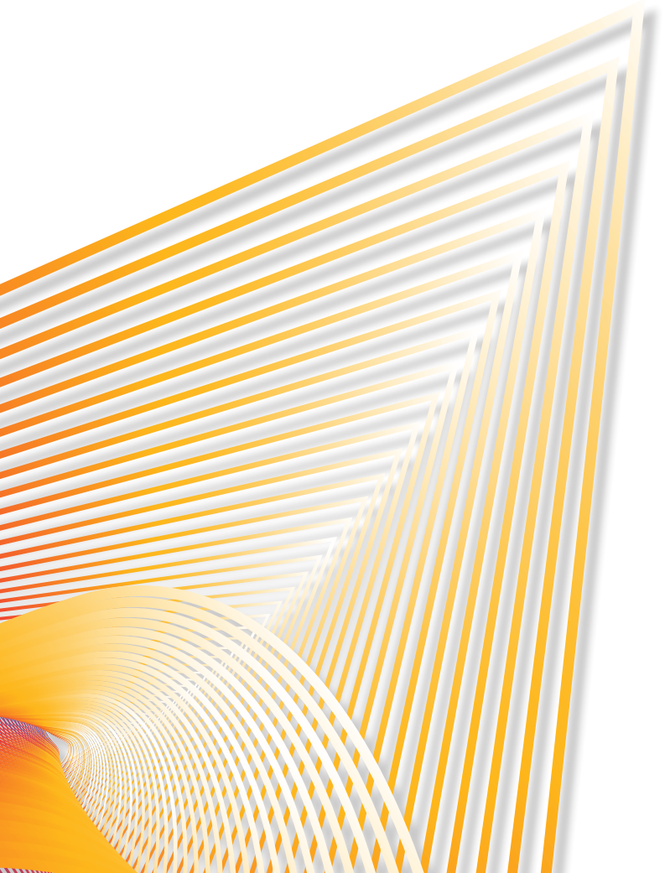
2025

# TABLE OF CONTENTS





# UNPRECEDENTED CHANGE FOR EMBEDDED SOFTWARE



The world of embedded software is changing at a faster pace than ever before. Our research drills into this new reality and finds two major stories unfolding at once. The first is the story of artificial intelligence—a massive adoption of AI tools but paired with dangerously lagging governance. The second is the story of the software supply chain becoming a core business function with the maturation of Software Bills of Materials (SBOMs) into a mainstream commercial requirement.

This report is a guide to understanding those two stories as well as the current state of embedded software development. We surveyed 785 professionals in the trenches—the developers, managers, and security pros who build the embedded software that runs our world—to give you a real, data-driven look at what's happening.

# Why You Should Read This Report

"The State of Embedded Software Quality and Safety 2025" report is a field guide to the realities of embedded development. Whether you're managing the budget or writing the code, the data presented here is directly relevant to your work.

## For Executive Leaders

This report is about risk and competitive advantages. Our findings on "shadow AI" and the governance gap aren't technical problems; they are unmanaged business risks that create legal exposure and threaten your intellectual property. The data on SBOMs becoming a commercial requirement is a clear signal about what it now takes to win and keep customers. And the insights into the changing skillset of developers can directly inform your hiring, training, and strategic planning. This report gives you the information you need to ask the right questions from your technical teams and make informed decisions about where to invest.

## For Hands-on Coders

This report is about your job and your skills. The data showing Python overtaking C++ is a headline you can't ignore; it's a strong indicator of what coding skills are increasingly in demand. The findings on memory-safe languages and the "shift-everywhere" approach to security show how the definition of "good code" is changing. And the data on the manager/engineer perception gap is validation that the pressure you're feeling is real. Use this report to understand the trends shaping your career and advocate for the tools and processes you need to do your job effectively and securely.

## Survey Methodology

This report is designed to give you a clear, data-backed understanding of the current state of the embedded software industry. We break down the key trends, the real-world challenges, and the strategic shifts separating the leaders from the laggards. Our goal is to give you the insights you need to make smart decisions in a landscape that's changing by the minute.

The data comes from a comprehensive survey conducted by the international market research firm Censuswide in June 2025, making "The State of Embedded Software Quality and Safety" report findings the most timely you're likely to find. We collected responses to 16 key questions from 785 professionals who live and breathe embedded software every day.

Throughout the report you'll see markers such as this: **[Q12]**. A marker indicates that the data preceding it was derived from one of those 16 questions. The survey's full questions and responses can be found in the Appendix.

A quick look at who we talked to

- **Geographies:** This report is a global snapshot, with responses from the U.K., U.S., Singapore, Finland, France, Germany, Japan, and China.
- **Roles:** We covered the full spectrum of industry roles, from hands-on coders and their team leads to senior technical management, security and compliance officers, and those working on AI and emerging tech.
- **Companies:** Respondents came from organizations of all sizes and across all major industry verticals, including automotive, tech, manufacturing, and MedTech, giving us a well-rounded view of the embedded software ecosystem.

# EXECUTIVE SUMMARY

For the C-suite, the takeaway from our data is simple: The ground beneath your feet is shifting. The tools your teams use, the skills they need, and the risks they face are being completely redefined. The AI revolution and the formalization of supply chain security are your new operational reality. Ignoring these evolving changes is not an option if you care about your competitive position, your IP, or your products' security.

Here are the six key findings you'll find in this report.

***If you can't show what's in your software, you're at a competitive disadvantage.***

## Top Six Key Findings

- **The Stakes Are High:** The top concern among our respondents around software being released with defects was the possible safety/environment impact (19.62%) [Q2].
- **AI Is Everywhere, but the Guardrails Are Missing:** AI adoption is in full swing. A staggering 89.3% of respondents say their companies are already using AI coding assistants, and 96.1% note that their companies are building open source AI models directly into their products [Q10, Q11].

  But here's the problem: Governance is lagging far behind. Over 21% of organizations admit that they aren't confident they can stop AI from injecting new flaws and issues into their code. That's not just a gap; it's a gaping hole in the development life cycle [Q12].
- **SBOMs Aren't Just for Regulators Anymore:** SBOMs are no longer a checkbox for government contracts; they are a commercial demand. Over 70% of organizations now must produce an SBOM [Q15].

  And who's asking for those SBOMs? Your customers. "Customer or partner requirements" (39.4%) is the biggest driver, beating out "industry regulation" (31.5%). The market has spoken: If you can't show what's in your software, you're at a competitive disadvantage. [Q15].
- **The Embedded Developer Is a New Breed:** The job description for an embedded developer is being rewritten. While C languages are still the cornerstone of embedded systems, it's increasingly about memory-safe languages, with 80.4% of companies having already adopted them [Q9].
- **"Shadow AI" Is the Threat You May Not Be Tracking Sufficiently:** What's worse than a risk you know about? One you don't. A significant 18% of companies know their developers are using AI tools against company policy [Q10]. Shadow AI is a massive, unmanaged risk vector for your IP and your security posture.
- **Your Managers and Engineers Are Living in Different Realities:** Think your projects are running smoothly? Your engineers would probably disagree. While over 85% of VPs and directors are optimistic about on-time, on-quality releases, only 64% of the hands-on developers share that sentiment [Q7].

  Managers see a successful release; engineers see the painful compromises and technical debt they incurred to hit the deadline. This isn't just a communication problem; it's a fundamental disconnect about risk and quality.

**Q2** If software is released with defects that result in a major event (e.g., serious malfunction, data breach, etc.), what is your largest concern, if anything?

**Q10** Are your developers using AI-powered code assistants to help write code?

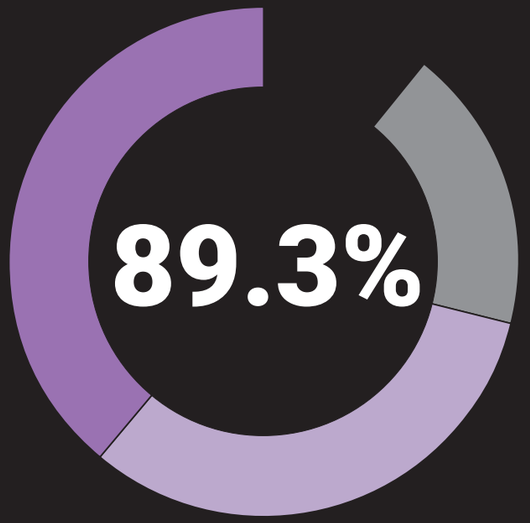
**Q11** Are you using any open source AI models (e.g., from Hugging Face) in the software you build?

**Q12** How confident are you that you have the processes and tools in place to ensure AI-generated code doesn't introduce security vulnerabilities or other issues?

**Q15** Is your business required to produce a Software Bill of Materials (SBOM) to meet customer requirements or industry regulations?

**Q9** Are your developers writing code (or planning to begin writing code) using one or more "memory safe" programming languages (e.g., Rust, Go, C#, Swift, Java, or Python)?

**Q7** How successful is your organization in releasing software on time that meets your coding standards?




Some developers are permitted to use AI assistants [39.0%]

All developers are permitted to use AI assistants [32.3%]

Used against company policy ("Shadow AI") [18.0%]

**Breakdown of AI Assistant Usage [Q10]**

# THE AI REVOLUTION IN EMBEDDED SYSTEMS

Of all the changes sweeping the industry, one stands out as the most disruptive: artificial intelligence. Developers are adopting AI tools at a breakneck pace, but the rules and safeguards needed to manage this new power are dangerously behind. This is the story of that adoption and the critical governance gap it has created.

## Unprecedented Adoption of AI

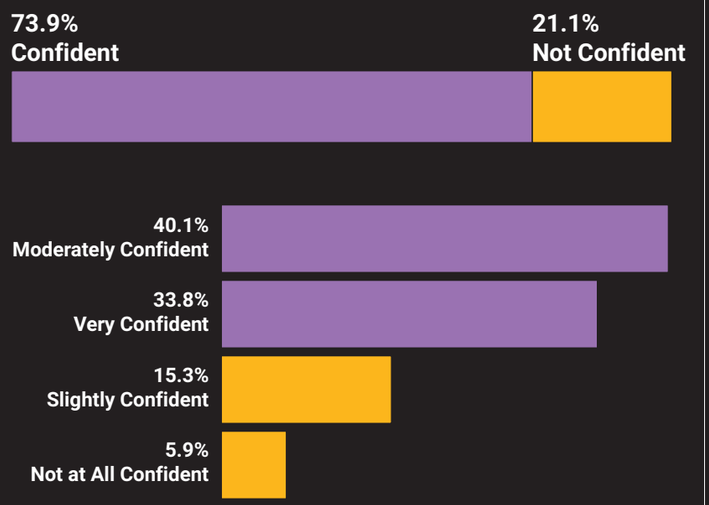
**Q10** Are your developers using AI-powered code assistants to help write code?

**Q11** Are you using any open source AI models (e.g., from Hugging Face) in the software you build?

Make no mistake, the debate about whether to use AI in embedded development is over. It's here, it's established, and it's everywhere. This isn't an emerging trend; it's a shift in how software is made.

- **It's Ubiquitous:** A stunning 89.3% of companies are already using AI-powered coding assistants [Q10]. Think about that. AI is now as much a part of the developer's toolkit as a compiler or a debugger. It's fundamentally altering the day-to-day work of creating, fixing, and optimizing code.
- **It's Built-in:** This isn't just about developer aids. AI is being woven directly into the fabric of embedded products. According to our respondents, 96.1% of companies are using open source AI models in the software they ship. Nor are these trivial uses. AI is powering core functions like data processing (37.3%), computer vision (35.9%), and process automation (34.8%) [Q11].
- **It's Unsanctioned; the Shadow AI Problem**: Eighteen percent of companies know their developers are using AI tools against official policy [Q10]. This isn't just a minor compliance issue; it's an unmanaged risk. Just as with human-developed code, AI-generated code needs to be thoroughly tested for potential security flaws, license violations, and IP issues.

***18% of organizations are aware that their developers use AI tools even when it is against stated company policy.***




**Confidence in Securing AI-Generated Code [Q12]**

# THE GOVERNANCE GAP: CONFIDENCE LAGS BEHIND ADOPTION

AI tools are everywhere, but the policies and processes to control them are lagging dangerously behind. AI risk management is a new, significant category of business risk that most companies are only beginning to grapple with.

- **The Security Oversights:** Are you sure your AI-generated code is safe? More than one in five companies (21.1%) aren't sure of that at all. That group includes 15.3% that are only "slightly confident," and another 5.9% that are "not at all confident" in their ability to keep AI from introducing vulnerabilities [Q12]. For a fifth of the embedded software industry, the race to adopt AI has completely outpaced the ability to secure it.
- **The Ticking Time Bomb of IP Risk:** The confidence gap around intellectual property is just as bad. Nearly 20% of organizations admit that they're not confident they can manage the open source license risks that come with AI-generated code [Q13]. Since many AI models are trained on the entire internet's open source code, they are veritable license-laundering machines. Without proper checks, your next AI-assisted feature could come with a potential GPL-v3 lawsuit attached.
- **A Failure in Risk Management:** When your teams are shipping code from unvetted, AI-powered sources without automated security and license scanning, you are actively introducing unknown risk into your products. The speed gained today is technical debt—and legal liability—being deferred to tomorrow.
- **The Confidence Chasm Between Security and Dev**: Who is most worried about this? The people writing the code. There's a telling chasm between the confidence of security teams and the developers on the front lines. Over 44% of embedded software engineers feel unequipped to handle the license risks from AI. Yet only about 14% of their counterparts in product security share that lack of confidence [Q13].

**Q12** How confident are you that you have the processes and tools in place to ensure AI-generated code doesn't introduce security vulnerabilities or other issues?

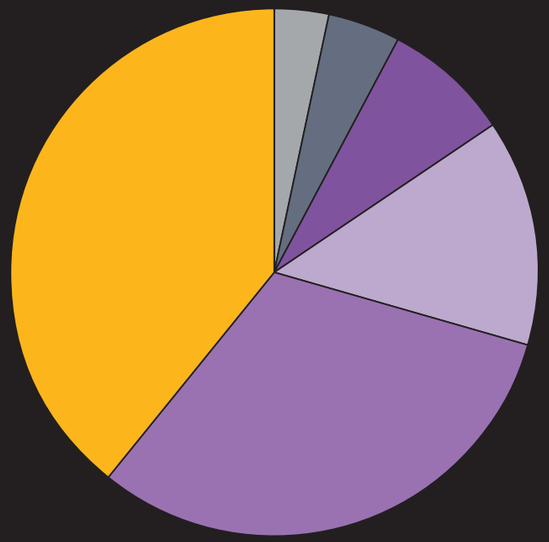
**Q13** How confident are you in your ability to ensure AI-powered code assistants don't introduce open source code subject to license obligations that could put your IP at risk?

***The disconnect between rapid, often unsanctioned adoption and lagging governance creates a significant new category of business risk.***

# THE MATURATION OF SOFTWARE SUPPLY CHAIN MANAGEMENT

While the rush to AI adoption can feel like the Wild West, a different story is unfolding in the software supply chain. What was once a niche security concern has become a mainstream business requirement with transparency, governance, and risk mitigation now the price of admission. This section looks at how companies creating embedded systems are embracing SBOMs as a commercial necessity.

***More than half of all companies are actively scanning for license obligations, both in open source components and in code snippets.***



- Yes, we currently produce SBOMs to meet customer or partner requirements [39.4%]
- Yes, we currently produce SBOMs to comply with an industry regulation [31.5%]
- No, but we do produce SBOMs for a different reason [13.8%]
- No, we don't currently produce SBOMs, but are planning to in the future [7.9%]
- No, we have no requirement to produce SBOMs [4.2%]
- I do not have enough visibility into our SBOM process [3.3%]

**Primary Drivers for SBOM Production [Q15]**

## Open Source Software in Embedded Systems

Open source isn't just "a part" of embedded software. It's often a major part of the code. It's in the languages developers choose as well as the components they use, and it influences the way their organizations approach code quality and security.

- **SCA Is Now SOP:** You can't depend on open source without knowing what's in it. Software composition analysis (SCA) is now a standard practice, with scans happening at every stage of the pipeline: with every build (39.1%), on every pull request (38.9%), and even right inside the developer's IDE (34.9%) **[Q3].** This trend is far beyond shifting left; it's a "shift-everywhere" strategy that accepts the need for constant visibility.
- **License Compliance Is No Longer an Afterthought:** More than half of all companies are actively scanning for license obligations in their main components (51.0%) and even in the tiny code snippets that developers copy and paste (54.4%) **[Q5]**. As always, protecting the company's IP needs to be a major development concern.

**Q3** What types of testing are you running to identify code quality issues, defects, and vulnerabilities in your software code?

**Q5** What measures are you taking to avoid software license conflicts or to reduce risks to your intellectual property, if any?

**Q15** Is your business required to produce a Software Bill of Materials (SBOM) to meet customer requirements or industry regulations?

**Q16** How confident are you in your company's ability to produce a complete and accurate Software Bill of Materials (SBOM)?

## SBOMs Become a Commercial Imperative

SBOMs have made the leap from a government compliance requirement to a must-have commercial deliverable.

- **It's a Mainstream Requirement:** The question is no longer *if* you need an SBOM, but *when*. A full 70.8% of companies now say that producing an SBOM is a requirement for their business **[Q15]**.
- **The Market Is in the Driver's Seat:** For years, the push for SBOMs was a top-down, government-led effort. Not anymore. The single biggest reason companies produce SBOMs today is to meet "customer or partner requirements" (39.4%) **[Q15]**. The market is now demanding transparency, making SBOMs a tool for competitive advantage.
- **Embedded Software Producers Lead in SBOMs:** Despite the challenges of creating an accurate SBOM, 80.4% of companies are confident they can produce a complete and accurate one when asked **[Q16]**. This suggests that the tooling and processes for SBOM generation are maturing for embedded system producers more rapidly than in other markets.

***The single biggest driver for producing an SBOM is to meet customer or partner requirements.***

# THE PEOPLE AND PROCESSES OF MODERN EMBEDDED DEVELOPMENT

Technology and supply chains are only part of the story. This section digs into the data on the life of the embedded developer, the constant battle between speed and quality, and the messy reality of compliance in 2025.

## Navigating Speed vs. Quality

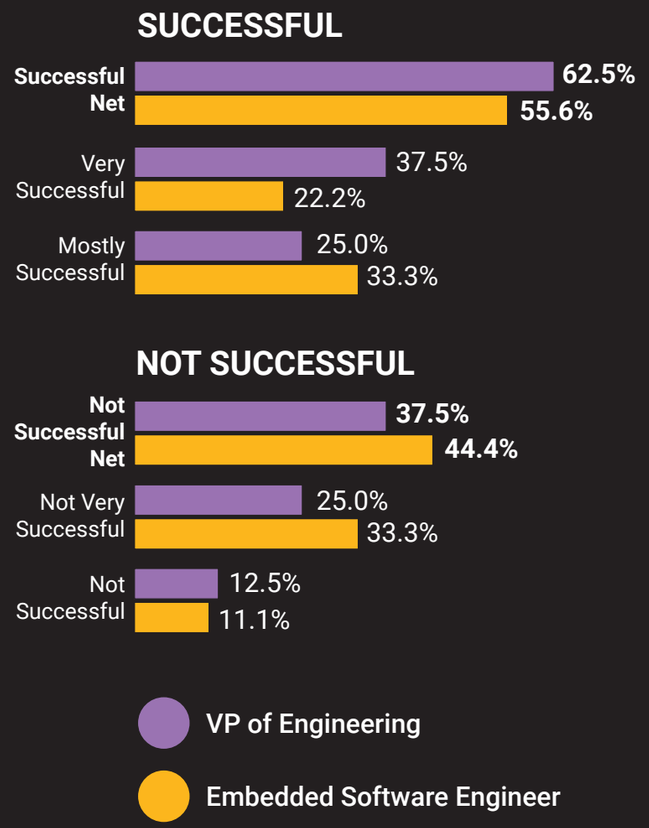
The age-old battle in software—"get it done fast" versus "get it done right"—is alive and well in the embedded world.

- **The Necessary Evil of Compromise:** Do teams feel successful? Mostly. About 71% report success in meeting their goals. But the devil is in the details. Nearly 40% of all respondents admit they are only "mostly successful" because they "sometimes need to compromise" on quality to hit a deadline [Q7].
- **The More Things Change, the More They Stay the Same:** For all the talk of new tools and processes, the biggest headaches for developers are the same as ever. The #1 challenge is the sheer "complexity of our software, hardware, and/or systems" (18.7%), followed immediately by "tight release timelines" (18.1%) [Q8]. As embedded products get smarter and more connected, the pressure on the teams building them only intensifies.

**Q7** How successful is your organization in releasing software on time that meets your coding standards?

**Q8** What is the biggest challenge to eliminating critical defects and vulnerabilities in your software projects?

***A stark perception gap:* 86% of CTOs feel their projects are successful, while only 56% of hands-on embedded software engineers agree.**




**Release Success Perception: Management vs. Engineers [Q7]**

- **The Manager/Engineer Reality Gap:** The data shows that managers and their engineers are living in two different worlds. Ask a CTO if their projects are successful and 86% will say yes. Ask an embedded engineer on the front lines and that number plummets to 56% **[Q7]**.

  This isn't merely a difference of opinion. It's a chasm in perception. A *manager* sees a product shipped on time and calls it a win. An *engineer* sees the shortcuts, the accepted risks, and the technical debt kicked down the road and calls it a compromise. The perception gap is a significant source of hidden risk in many organizations.

## The Fragmented Compliance Landscape

For an industry so focused on safety and security, compliance among embedded software companies is surprisingly fragmented. There is no single standard that rules them all.

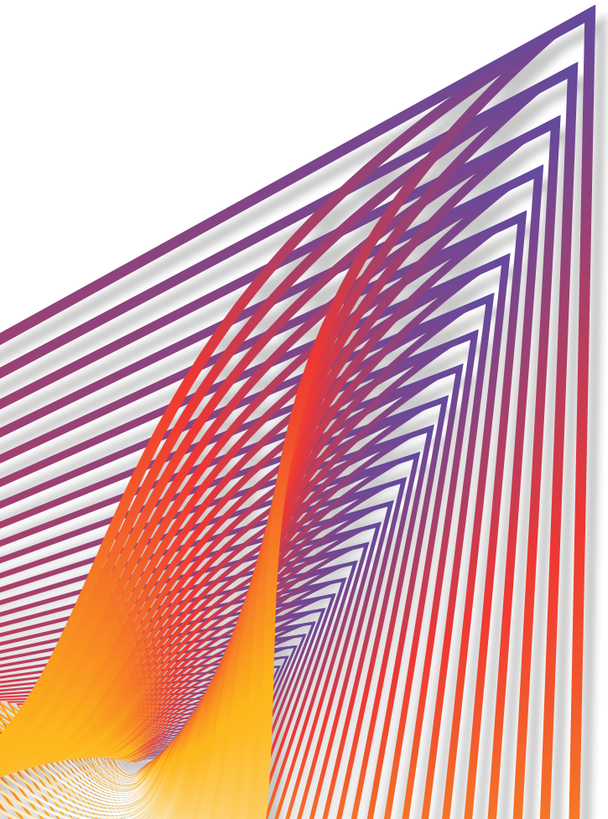
- **Internal Standards Are King:** When it comes to the rules developers have to follow, no single public standard dominates. The most common source of authority is a company's own "internal coding standards," cited by 22.2% of respondents. That's just as common as established frameworks like MISRA C/C++ (22.0%) or ISO 26262 (21.2%) **[Q14]**.
- **The "Build-Your-Own" Compliance Framework:** What does this fragmentation mean? It means that smart companies are not just picking a single standard and following it unquestioningly. They're creating their own hybrid compliance frameworks. They are cherry-picking the controls that matter most from a variety of standards—CERT C, CWE, ISO—and combining them into a custom policy that fits their specific products and risk appetite. This, in turn, means they need security and testing tools that are flexible enough to enforce their unique, blended rule sets.

**Q7** How successful is your organization in releasing software on time that meets your coding standards?

**Q14** What standards or regulations must your software adhere to before it's released?

***Embedded software organizations need security and testing tools that are flexible enough to enforce their unique, blended rule sets.***

# RECOMMENDATIONS AND OUTLOOK



## Thriving in the New Embedded Paradigm

The embedded industry is being remade by two powerful forces: the integration of AI, the formalization of the software supply chain. Our report shows that navigating this new world isn't about buying one new tool or adopting one new process. It's about a holistic strategy that connects your technology, your people, and your governance. The companies that get this right will innovate faster and more securely. Those that don't will be buried under a mountain of risk from unmanaged AI, opaque supply chains, and an outdated workforce.

## Actionable Recommendations

Here are our recommendations for the key players in your organization.

### For Technical Leaders (Developers, Architects)

- **Treat AI Assistants Like Talented, Unreliable Interns:** Ensure that you're ready for the testing scale that AI will require. AI-generated code can be a great first draft, but it's still a first draft. It needs rigorous, human-in-the-loop validation. Review every line. Test every function. Verify everything. Assume nothing.
- **Demand Better, Integrated Tooling:** Your biggest challenges are complexity and time. Don't accept security tools that slow you down. Advocate for modern static application security testing (SAST) and SCA tools that live where you live—in your IDE and your CI/CD pipeline—to find and fix problems early.

### For Managers (Directors, VPs)

- **Write Your AI Policy Yesterday:** A probable 18% of your team using shadow AI is a direct threat to your business. You need a formal AI usage and governance policy, and you need it now. Define what's allowed, what's not, and how AI use will be monitored.
- **Your Team's Skills Are Getting Out-of-Date; Fix the Problem:** The data is clear: The skills that got you here won't keep you here. Use the findings in this report to justify immediate investment in training for memory-safe languages, and just as importantly, for the secure and effective use of AI.
- **Stop Treating Security Like a Cost Center:** The gap between your perception of success and your engineers' reality is a business risk. Use this data to reframe your security budget. It's not an expense; it's a critical investment in risk mitigation that enables your team to move faster, safely.

### For Security and Compliance Professionals

- **Audit Your Tools for AI-Readiness:** Can your current toolchain identify and analyze code generated by a large language model (LLM)? Can it scan the open source AI models your teams are downloading from Hugging Face for vulnerabilities and license issues? If the answer is no, you have a critical visibility gap.
- **Update Your Threat Models:** Your risk register is obsolete if it doesn't include AI-specific threats. You need to be actively modeling the risks from shadow AI, AI-introduced vulnerabilities, and the complex "license-laundering" problem of modern code generation.
- **Weaponize the SBOM:** Your customers are demanding SBOMs. Don't just treat this as a compliance task. Use SBOMs as a strategic asset—information that accurately reflects all dependencies (including transitive ones), provides correct versioning information, and is useful for downstream risk management activities such as vulnerability management and incident response.

## Future Outlook

The trends we've identified are only going to accelerate. By 2026, we expect to see a boom in AI governance tools as the industry rushes to control its use. This will create a clear and widening skills gap between developers who can master these new tools and those who can't. And finally, the SBOM will cease to be a "nice to have" and will become a standard, non-negotiable line item in most B2B contracts, cementing its role as a fundamental part of the software economy.

# HOW BLACK DUCK CAN HELP

## From Insight to Action in the New Embedded Paradigm

The findings in this report paint a clear picture of an industry during a profound transformation. The disruptions of ubiquitous, AI-assisted development, the adoption of AI models, and the maturation of software supply chain management have created a new, more complex risk landscape for embedded systems. The data indicates that the old models for securing software—siloed tools, manual processes, and security as a final gate—are no longer sufficient. Navigating this new paradigm requires moving beyond disconnected point solutions to an integrated, platform-based approach that addresses security, quality, and compliance.

This is building trust in the software that powers our world. Doing so requires a strategic partner with the pedigree, technology, and deep domain expertise to manage risk at the speed and scale that modern embedded development demands. Black Duck offers the industry's most comprehensive and trusted portfolio of application security solutions, with an unmatched track record of helping organizations secure their software, integrate security efficiently, and innovate safely with new technologies. The following sections detail how Black Duck's solutions directly address the key challenges identified in this report for both executive leaders and the hands-on developers building the next generation of embedded systems.

# FOR EXECUTIVE LEADERS: TRANSFORMING SYSTEMIC RISK INTO COMPETITIVE ADVANTAGE

For the C-suite and senior management, the challenges highlighted in this report—unmanaged AI, opaque supply chains, and the widening gap between management perception and engineering reality—are systemic business risks that threaten intellectual property, create legal exposure, and erode competitive advantage. Black Duck provides the enterprise-level visibility, governance, and control necessary to transform these risks into strategic opportunities.

## Closing the AI Governance Gap: Taming Shadow AI and Protecting IP

### THE PROBLEM

This report reveals a stark paradox: AI adoption is nearly universal, with 89.3% of organizations using AI assistants [Q10], but the governance required to manage it is dangerously lagging. Over 21% of organizations admit they are not confident they can prevent AI from introducing new flaws into their code [Q12]. More alarmingly, 18% of companies know their developers are using unsanctioned shadow AI tools against company policy [Q10], creating a massive, unmanaged risk vector for both security and intellectual property.

**Q10** Are your developers using AI-powered code assistants to help write code?

**Q12** How confident are you that you have the processes and tools in place to ensure AI-generated code doesn't introduce security vulnerabilities or other issues?

### THE BLACK DUCK SOLUTION

The AI governance gap is fundamentally a data and provenance problem. You cannot govern what you cannot see. Code generated by LLMs is of unknown origin, creating two distinct and critical business risks: a composition risk (*What is this code and what are its license obligations?*) and a quality risk (*Is this code secure and reliable?*). A security-only approach that neglects composition risk leaves a massive IP liability on the table. Black Duck's portfolio is uniquely positioned to solve the complete AI governance problem by addressing both.

**Black Duck® SCA:** To counter IP and license risk, Black Duck SCA provides deep analysis of code. Because LLMs are trained on vast public codebases, they can inadvertently "launder" code from projects with restrictive licenses, inserting it into proprietary software.

Further, Black Duck's powerful **snippet analysis** technology matches small pieces of AI-generated code back to their original open source projects, identifying their true licenses and associated obligations. This capability directly mitigates the legal and financial exposure that is a top concern for executive leadership.

The Black Duck snippet analysis API is a scalable solution for integrating snippet analysis into automated workflows, such as CI/CD pipelines. Developers send raw source code to the API. Black Duck then hashes this source code and compares it against its KnowledgeBase™, which contains a vast repository of open source code and associated metadata. If matches are found, the API returns the results, typically in JSON format. This output includes details such as the matching open source component and version, license name and type, and the location of the matched snippet within the source file.

# 71%

**of users reported at least a 10% reduction in time spent finding and fixing code issues after implementing Coverity.**

Survey conducted and verified by UserEvidence.

**Coverity® Static Analysis:** To address security and quality risk, Coverity Static Analysis analyzes AI-generated code for critical defects and vulnerabilities. AI tools can inherit and replicate low-quality code from their training data, introducing flaws that are difficult to spot with manual review. Coverity's deep, accurate static analysis scans this code before it is integrated into the main codebase, ensuring that the velocity gained from using AI does not come at the expense of security, safety, or quality.

Coverity excels at identifying quality and security issues within C/C++ code. Its capabilities extend to supporting a wide range of C/C++ frameworks, compilers, and adherence to coding standards such as MISRA and CERT C/C++.

***Coverity has been really valuable for our team because it helps us catch security and quality issues early in the development process, before they become costly to fix. It integrates well with our CI/CD setup, so the scans happen automatically without slowing anyone down. What stands out is the low false positive rate and the clear guidance it gives developers, which makes it much easier to actually fix issues."*** —Senior InfoSec Engineer

**Complementing Coverity's capabilities, Black Duck SCA provides industry-leading C/C++ scanning to accurately identify the open source dependencies and libraries used in C/C++ applications.**

## Mastering the Software Supply Chain: The Strategic SBOM

### THE PROBLEM

SBOMs have transitioned from niche compliance to commercial imperative. This report finds that over 70% of organizations are now required to produce an SBOM, with customer and partner requirements (39.4%) being the single largest driver—outpacing even industry regulation (31.5%) [Q15]. The market-driven demand signals a fundamental shift: Customers now require deep transparency into their software supply chains.

**Q15** **Is your business required to produce a Software Bill of Materials (SBOM) to meet customer requirements or industry regulations?**

### THE BLACK DUCK SOLUTION

Black Duck SCA is the definitive solution for creating the *accurate, comprehensive, and dynamic* SBOMs that the market and regulators now demand.

**Unmatched Visibility:** A truly accurate SBOM must go far beyond what is declared in a package manager. Black Duck's multifactor detection capabilities provide this depth by combining dependency analysis, snippet analysis, and—critically, for the embedded space—binary analysis. For embedded systems, where access to source code is often limited or impossible, the ability to analyze compiled binaries is non-negotiable for achieving a complete and accurate inventory.

Complementing Coverity's capabilities, Black Duck SCA provides industry-leading C/C++ scanning to accurately identify the open source dependencies and libraries used in C/C++ applications, even where there is no presence of package managers.

A complete and accurate SBOM also includes all application dependencies. This means that third-party and custom components should be included, as well as any open source libraries added by your development teams. While open source makes up much of modern applications, many development teams still rely heavily on vendor-supplied components like libraries, SDKs, drivers, and so on. If these components are not accounted for in the SBOM shipped with the finished application, complete visibility of supply chain risk has not been obtained.

With Black Duck SCA, teams can import third-party SBOMs to automatically map dependencies to known components, and it will create an "unrecognized dependency" entry for custom or commercial dependencies that aren't already present in the KnowledgeBase. SBOMs containing all open source, custom, and commercial dependencies can also be exported in SPDX or CycloneDX formats, to align with customer, industry, or regulatory requirements.

*The integrations into the CI/CD process, and Black Duck's own vulnerability research (Black Duck Security Advisories), give us an advantage to fix vulnerabilities that are published by our own team, instead of just working with CVEs."*

—Jullian Diaz, Secure Coding Officer, Banco General

**Tackling Transitive Dependencies:** This report's companion "Open Source Security and Risk Analysis" (OSSRA) report finds that 64% of open source components in a typical codebase are transitive dependencies—a primary source of hidden risk and license conflicts. Black Duck SCA excels at mapping these complex, multilevel dependency chains, ensuring that the SBOM is a true reflection of all the software in the final product, not just the components developers added directly.

The market's demand for SBOMs reflects a deeper need than simple compliance. A static, one-time file may satisfy a checkbox, but what customers truly require is ongoing assurance. The SBOM must be a living, dynamic asset that reflects the real-time risk posture of the software throughout its life cycle. Black Duck SCA enables this by creating a persistent Software Bill of Materials within its platform that is continuously monitored against the **Black Duck KnowledgeBase**, the industry's most comprehensive database of open source components, vulnerabilities, and license data. When a new vulnerability is discovered in a component—even months or years after a product has shipped—Black Duck SCA alerts your organization of the issue. This capability transforms the SBOM from a static compliance artifact into a strategic tool for proactive, life cycle–long risk management, which is what customers and regulators are truly demanding.

*I think the Black Duck KnowledgeBase is the most comprehensive in the industry, and Black Duck Security Advisories are also very helpful in speeding up remediation."*

—Ned Krtolica, RelOps Engineering Manager, Dassault Systemes

**59%**

**of users decreased their security defect rate by at least 25% after implementing Black Duck solutions.**

Survey conducted and verified by UserEvidence.

## Bridging the Management/Engineering Gap

### THE PROBLEM

The data in this report reveals a stark chasm in perception between management and the engineers on the front lines. While 86% of CTOs feel their projects are successful, only 56% of hands-on embedded engineers agree [Q7]. This is not a communication problem; it is a systemic business risk born from disconnected tools and a lack of a shared, objective view of reality. According to Black Duck's "Global State of DevSecOps" report, the problem is compounded by tool proliferation, with 82% of organizations using between 6 and 20 different security tools, creating noise, inefficiency, and a fragmented view of risk.

**Q7** **How successful is your organization in releasing software on time that meets your coding standards?**

### THE BLACK DUCK SOLUTION

The **Black Duck Polaris™ Platform** provides the single, correlated source of truth needed to bridge this gap.

**Centralized Visibility:** Polaris integrates the findings from Black Duck's entire portfolio and other third-party tools—SCA, SAST, dynamic application security testing (DAST), and more—into a unified dashboard. This gives executives a single pane of glass to view and manage application security risk across the entire enterprise, eliminating the silos created by disconnected point solutions.

**Role-Based Views:** A single source of truth does not mean a one-size-fits-all view. Polaris presents correlated data through the lens appropriate for each role. *Executives* receive high-level risk posture reports and compliance dashboards that map directly to business priorities. *Security managers* can create and enforce policies. *Developers* receive precise, actionable findings directly in the tools they already use. By ensuring that everyone is working from the same underlying data, the platform creates a shared understanding of risk and quality, directly closing the dangerous perception gap and enabling genuine, data-driven collaboration.

# FOR HANDS-ON DEVELOPERS: BUILDING HIGH-QUALITY SOFTWARE WITHOUT SACRIFICING SPEED

For hands-on developers, architects, and team leads, the core challenge is a battle against complexity and time. The pressure to deliver quickly is immense, but the increasing complexity of embedded systems means that quality, security, and safety cannot be compromised. Security testing that is slow, inaccurate, or that forces developers out of their established workflows is not just an inconvenience; it is a direct impediment to productivity. Black Duck's solutions are designed by and for developers, providing the speed, accuracy, and seamless integration needed to build secure, high-quality software without sacrificing velocity.

***Coverity has been impactful for our team, primarily due to its deep static analysis capabilities that help us identify complex bugs and security vulnerabilities early in the development life cycle. One of the key metrics we value is the reduction in security-related defects discovered post-deployment."***

—Karthik Shetty, DevOps Engineer, Philips

## Reliable Code That Lives in Your Workflow: Fast, Accurate, and Integrated

### THE PROBLEM

According to our survey data, the top challenges for developers are the "complexity of our software, hardware, and/or systems" (18.7%) and "tight release timelines" (18.1%) [Q8]. In a high-pressure environment, software testing must be a seamless and efficient part of the development process. Slow scans, high false-positive rates, and tools that require context-switching are major productivity killers.

**Q8** **What is the biggest challenge to eliminating critical defects and vulnerabilities in your software projects?**

### THE BLACK DUCK SOLUTION

Black Duck tools are engineered to integrate testing into the developer's natural workflow, providing fast, accurate feedback at the earliest possible moment.

**IDE and SCM Integration:** Both **Coverity** and **Black Duck SCA** provide plug-ins for popular IDEs and integrate directly with source code management (SCM) systems like GitHub, GitLab, and Bitbucket. This integration allows developers to receive real-time security and quality feedback as they write code, preventing entire classes of defects from ever being checked into the repository. The key benefit? This is the earliest, fastest, and cheapest point in the development life cycle to find and fix issues.

**95%**

**of Coverity users find that it supports their regulatory or compliance requirements (e.g., functional safety, audits, industry standards).**

Survey conducted and verified by UserEvidence.

**CI/CD Automation:** Scans can be fully automated within the CI/CD pipeline, triggered on every pull request or build. Using configurable policies, builds can be automatically passed, failed, or flagged based on the severity of findings, creating an automated quality and security gate that prevents critical issues from reaching the main branch or production.

**Comprehensive Language and Framework Support:** Our report highlights the evolving skillset of the embedded developer, with a decisive shift toward memory-safe languages and the rise of Python for AI/ML work. Black Duck provides deep support for the languages that matter most to modern embedded developers as well as over 200 frameworks and libraries, ensuring that developers have the right analysis for the right job.

## Enforcing Any Standard, Automatically

### THE PROBLEM

The embedded compliance landscape is uniquely fragmented. Our report finds that the most common source of authority is a company's own "internal coding standards" (22.2%), which are often a custom blend of controls from established frameworks like MISRA C/C++, ISO 26262, and CERT C [Q14]. This demands a flexible and configurable enforcement tool, not a rigid, one-size-fits-all solution.

**Q14** **What standards or regulations must your software adhere to before it's released?**

### THE BLACK DUCK SOLUTION

**Coverity** is built to provide the flexibility needed to enforce any standard, public or private.

**Configurable Checkers:** Coverity's static analysis engine features a comprehensive set of checkers that can be precisely tuned to match an organization's specific risk profile and compliance needs. Teams can easily enable or disable checkers to map directly to the controls required by any standard—from MISRA to a custom internal policy—providing the tailored enforcement required by the embedded industry.

**Compliance Reporting:** Coverity generates detailed reports that make it easy to track compliance with required standards and provide auditors and customers with tangible proof of adherence.

**71%**

**of users reported being more satisfied with how issue triage and remediation fits into their daily workflow after implementing Black Duck solutions.**

Survey conducted and verified by UserEvidence.

## Making AI a Superpower, Not a Liability

### THE PROBLEM

Developers are embracing AI assistants to boost productivity, but they are also on the front lines of the associated risk. Over 44% of embedded software engineers feel unequipped to handle the open source license risks that come with AI-generated code [Q13]. They need a safety net that protects them and their company without hindering their ability to leverage these powerful new tools.

**Q13** **How confident are you in your ability to ensure AI-powered code assistants don't introduce open source code subject to license obligations that could put your IP at risk?**

### THE BLACK DUCK SOLUTION

Black Duck provides the automated guardrails that make it safe for developers to innovate with AI.

**Real-Time Snippet Analysis:** The **Black Duck open source snippet API** is a game-changing tool for secure AI-assisted development. It allows the small blocks of code generated by AI assistants to be analyzed for license and security issues in seconds. By integrating this API call into the SCM workflow—for example, as a GitHub Action on a pull request—developers get immediate feedback on the provenance and risk of the code they are about to commit. This effectively shifts compliance left, making it a seamless part of the coding process rather than a needed late-stage audit.

**Actionable Guidance for Faster Fixes:** Both Coverity and Black Duck SCA deliver clear, actionable remediation guidance directly to the developer within their existing tools. This not only accelerates the time to resolution for identified issues but also serves as a valuable learning tool, helping developers improve their coding practices over time.

**An AI Security Companion in Your IDE:** Code Sight™ IDE Plug-in with Black Duck Assist™ automatically scans code in real time as it is written by developers or generated by AI coding assistants like GitHub Copilot. The AI-powered assistant identifies security vulnerabilities and potential open source license violations, allowing developers to address those issues instantly.

The report's findings on shadow AI being introduced at the developer's desktop and the need for continuous SBOM monitoring after deployment prove that a shift left–only strategy is no longer sufficient. Risk is introduced, discovered, and must be managed across the entire software development life cycle, and in response, a modern strategy must shift everywhere. Black Duck's portfolio is designed for this reality, providing fast, accurate analysis at every stage: in the IDE (before commit), within the CI/CD pipeline (before merge), and through continuous monitoring of deployed applications (after release). Comprehensive, life cycle–wide coverage is the only way to effectively manage the continuous and varied risks of modern embedded software development.

# APPENDIX A: FULL SURVEY QUESTIONS

**1. What is your policy regarding which defects or vulnerabilities can be present in your software when it releases?**

| | |
|---|---|
| We ensure all critical defects and vulnerabilities are resolved before a release | 21.40% |
| We may release with some defects or vulnerabilities, as long as the software adheres to one or more industry standards (e.g., MISRA, ISO 26262, OWASP, etc.) before releasing | 21.15% |
| We aim to resolve all known defects and vulnerabilities before a release | 19.36% |
| We decide which defects and vulnerabilities need to be resolved on a case-by-case basis | 17.96% |
| We may release with some defects or vulnerabilities, as long as the software adheres to internal policies or coding standards before releasing | 17.07% |
| I don't have enough visibility into our release policies | 3.06% |

**2. If software is released with defects that result in a major event (e.g., serious malfunction, data breach, etc.), what is your largest concern, if anything?**

| | |
|---|---|
| Impact on safety or the environment | 19.62% |
| Cost of patching defects in the field | 19.36% |
| Damage to company reputation | 17.58% |
| Loss of intellectual property | 16.69% |
| Noncompliance with industry standards | 13.89% |
| Legal ramifications | 12.87% |

**3. What types of testing are you running to identify code quality issues, defects, and vulnerabilities in your software code?**

| | |
|---|---|
| Static code analysis: On SCM events (e.g., pull requests) | 41.27% |
| Static code analysis: Periodic (e.g., nightly) full-project scans | 40.89% |
| Open source dependency scans: With every build | 39.11% |
| Open source dependency scans: On SCM events (e.g., pull requests) | 38.85% |
| Open source dependency scans: In the IDE | 34.90% |
| Static code analysis: In the IDE | 33.63% |
| I don't have visibility into our software testing | 2.80% |

**4. Which actions are automatically triggered by violations of coding standards or policies, if any?**

| | |
|---|---|
| Alerting to upstream contributors (e.g., developers, engineers, architects) | 32.74% |
| Block promotion into staging/production | 32.23% |
| Prevent checking-in of code to SCM/repositories | 32.10% |
| Prevent addition of compiled assets into binary repositories | 31.97% |
| Assignment to developers via issue management workflows (e.g., Jira, Slack) | 31.72% |
| Prioritization for triage and remediation | 30.70% |
| Alerting to downstream stakeholders (e.g., security team, partners, customers) | 30.57% |
| Breaking the build | 23.69% |
| No actions or mechanisms are automated, all are manual based on test results | 3.18% |

## 5. What measures are you taking to avoid software license conflicts or to reduce risks to your intellectual property, if any?

| | |
|---|---|
| Keeping software development and testing on-premises | 54.39% |
| Scanning for license obligations associated with open source code snippets | 54.39% |
| Scanning for license obligations of open source components | 50.96% |
| I don't have visibility into how we're protecting intellectual property | 3.18% |
| We are not taking any measures | 1.40% |

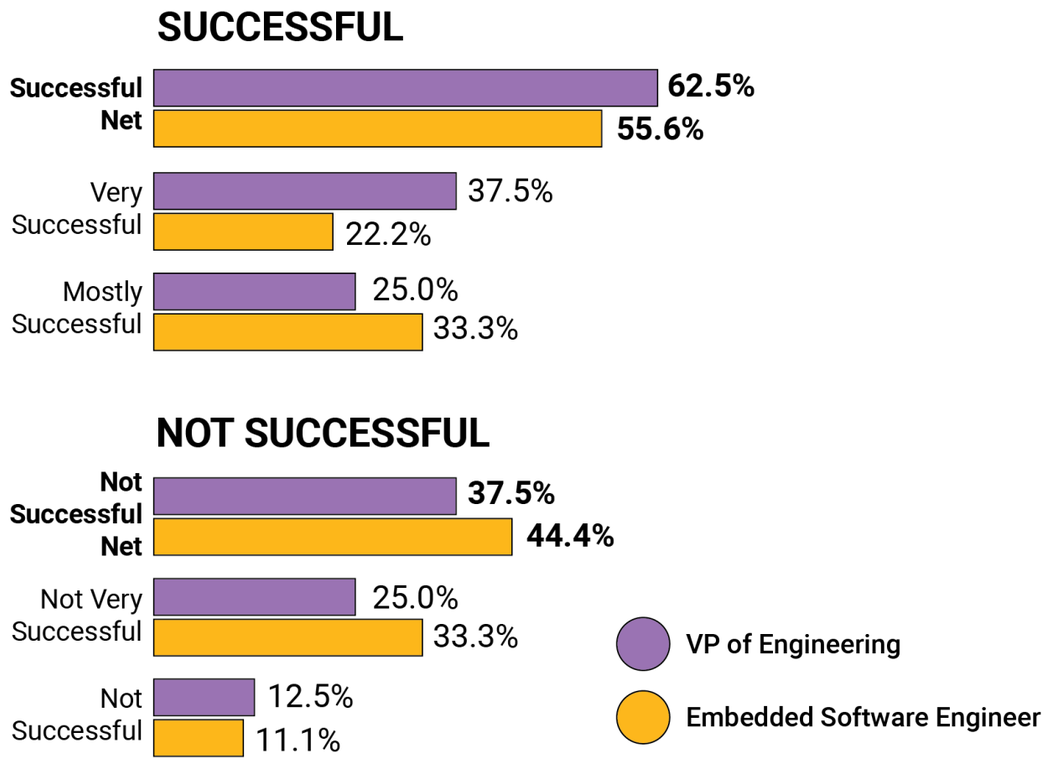
## 6. Which programming languages are your developers currently using?

| | |
|---|---|
| Python | 26.75% |
| C++ | 25.86% |
| Java | 21.53% |
| JavaScript | 21.40% |
| C# | 19.75% |
| C | 18.34% |
| Objective-C++ | 18.34% |
| Rust | 15.41% |
| Apex | 14.39% |
| CUDA | 14.39% |
| Ruby | 14.39% |
| TypeScript | 14.14% |

| | |
|---|---|
| Objective-C | 13.63% |
| VB.NET | 13.12% |
| Swift | 12.87% |
| PHP | 12.10% |
| Fortran | 11.72% |
| JSP | 11.59% |
| Scala | 10.83% |
| Go | 10.45% |
| Kotlin | 9.81% |
| Lua | 9.68% |
| Dart | 9.68% |

## 7. How successful is your organization in releasing software on time that meets your coding standards?

| | |
|---|---|
| Mostly successful, we usually release software on time that meets our coding standards, but we sometimes need to compromise | 39.62% |
| Very successful, we consistently release software on time that meets our coding standards | 31.34% |
| Not very successful, we sometimes release software on time that meets our coding standards, but we usually need to compromise | 15.03% |
| Not successful, we rarely release software on time that meets our coding standards, we almost always have to compromise | 10.06% |
| I do not have enough visibility into our releases and coding standards | 3.95% |



## 8. What is the biggest challenge to eliminating critical defects and vulnerabilities in your software projects?

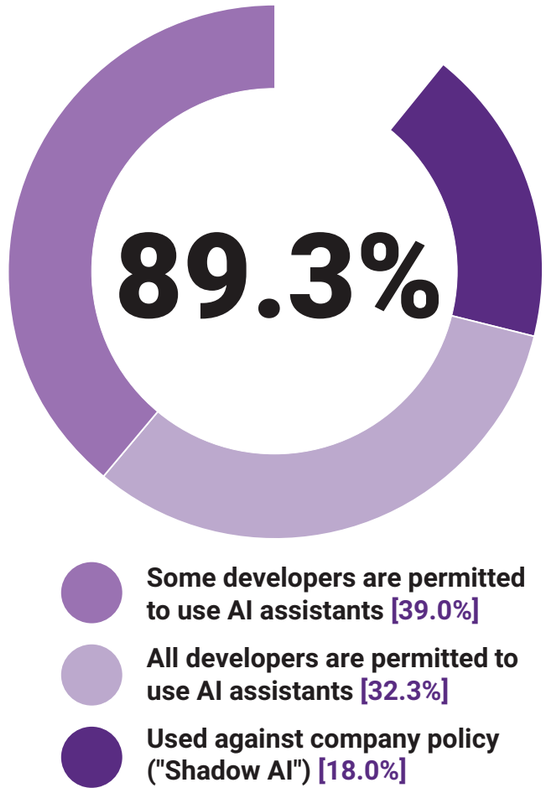
| | |
|---|---|
| Complexity of our software, hardware, and/or systems | 18.73% |
| Tight release timelines force us to compromise on which defects to resolve | 18.09% |
| External distractions making it hard to focus on resolving defects and vulnerabilities | 17.58% |
| It's not a priority to eliminate all critical issues | 14.65% |
| Insufficient testing or tools to identify issues | 14.39% |
| Lack of secure coding skills | 12.74% |
| I don't have visibility into challenges regarding coding defects | 3.82% |

## 9. Are your developers writing code (or planning to begin writing code) using one or more "memory safe" programming languages (e.g., Rust, Go, C#, Swift, Java, or Python)?

| | |
|---|---|
| We're already writing code in a memory safe language on new projects, and transitioning existing C++ projects to a memory safe alternative | 42.80% |
| We're already writing code in a memory safe language on new projects only | 37.58% |
| We're planning to begin writing code in a memory safe programming language in the near future | 13.50% |
| I don't have visibility into plans around programming languages | 3.18% |
| No, we have no plans to write code using a memory safe language | 2.93% |

## 10. Are your developers using AI-powered code assistants to help write code?

| | |
|---|---|
| Yes, but only certain developers/teams are permitted to, and do, use these tools | 38.98% |
| Yes, all developers are permitted to, and do, use these tools | 32.36% |
| Yes, while we do not allow the use of these tools, we are aware that some developers use them | 17.96% |
| No, developers are not permitted to, and do not, use these tools | 7.26% |
| I do not have enough visibility into development processes to know if these tools are used | 3.44% |

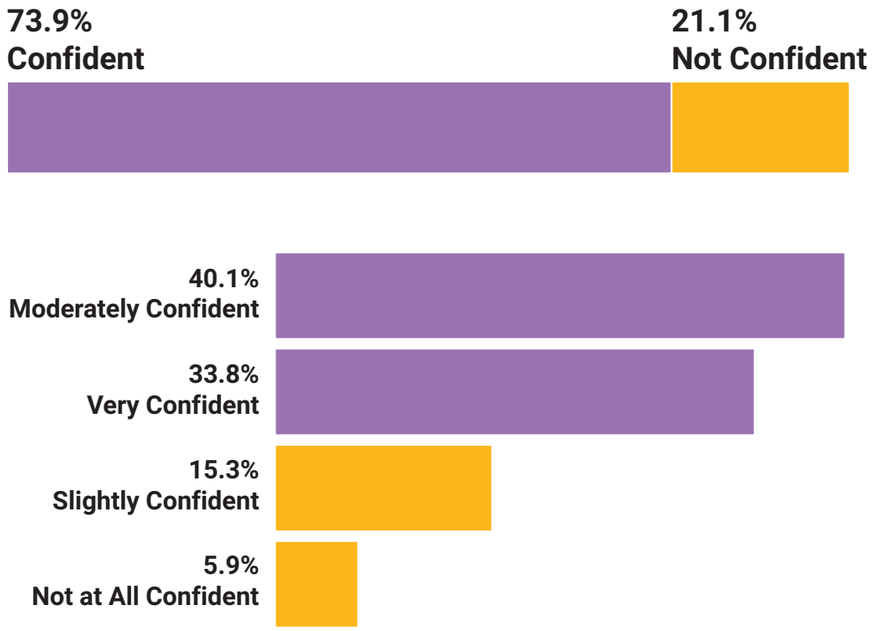


## 11. Are you using any open source AI models (e.g., from Hugging Face) in the software you build?

| | |
|---|---|
| Yes, for data processing and cleaning | 37.32% |
| Yes, for computer vision (e.g., image recognition) | 35.92% |
| Yes, for process automation | 34.78% |
| Yes, for natural language processing | 32.10% |
| Yes, for embedding pretrained models | 31.72% |
| Yes, for training custom models | 30.45% |
| Yes, for predictive analytics | 30.45% |
| I don't have visibility into usage of open source AI models | 2.93% |
| No, we are not currently using open source AI models | 1.02% |

## 12. How confident are you that you have the processes and tools in place to ensure AI-generated code doesn't introduce security vulnerabilities or other issues?

| | |
|---|---|
| Moderately confident we have sufficient policies and testing in place | 40.13% |
| Very confident we have sufficient policies and comprehensive testing in place | 33.76% |
| Slightly confident we have sufficient policies and testing in place | 15.29% |
| Not at all confident we have sufficient policies and testing in place | 5.86% |
| I do not have enough visibility into our processes to manage and secure AI-generated code | 3.31% |
| This is not a priority at this time, as using AI-generated code is against company policies | 1.66% |



## 13. How confident are you in your ability to ensure AI-powered code assistants don't introduce open source code subject to license obligations that could put your IP at risk?

| | |
|---|---|
| Very confident we have the policies and tools in place to identify all open source components and code snippets | 39.75% |
| Moderately confident we have the policies and tools in place to identify all open source components and code snippets | 38.85% |
| Slightly confident we have the policies and tools in place to identify all open source components and code snippets | 14.27% |
| Not at all confident we have the policies and tools in place to identify all open source components and code snippets | 4.08% |
| I do not have enough visibility into the policies and tools to identify all open source code in our software | 3.06% |

## 14. What standards or regulations must your software adhere to before it's released?

| | |
|---|---|
| CERT C/C++/Java | 24.46% |
| Internal coding standards | 22.17% |
| MISRA C/C++ | 22.04% |
| ISO 26262 | 21.15% |
| ISO/SAE 21434 | 20.89% |
| OWASP Top 10 | 19.62% |
| EU Cyber Resilience Act | 19.11% |
| CWE Top 25 | 18.47% |
| FDA Regulations | 18.34% |
| DO-326A/ED-202A | 18.22% |
| AUTOSAR | 18.09% |
| IEC 62443 | 17.07% |
| DO-178C | 15.16% |
| DISA STIG | 14.52% |
| PCI DSS | 14.01% |

## 15. Is your business required to produce a Software Bill of Materials (SBOM) to meet customer requirements or industry regulations?

| | |
|---|---|
| Yes, we currently produce SBOMs to meet customer or partner requirements | 39.36% |
| Yes, we currently produce SBOMs to comply with an industry regulation | 31.46% |
| No, but we do produce SBOMs for a different reason | 13.76% |
| No, we don't currently produce SBOMs, but are planning to in the future | 7.90% |
| No, we have no requirement to produce SBOMs | 4.20% |
| I do not have enough visibility into our SBOM process | 3.31% |

## 16. How confident are you in your company's ability to produce a complete and accurate Software Bill of Materials (SBOM)?

| | |
|---|---|
| Moderately confident we have tools in place to produce an SBOM, but aren't sure of its accuracy | 40.25% |
| Very confident we have the knowledge and tools in place to produce an accurate SBOM | 40.13% |
| Not confident we are unsure of what's required to build an accurate SBOM | 11.08% |
| Not a requirement to produce an SBOM at this time | 4.84% |
| I do not have enough visibility into our SBOM process | 3.69% |

# APPENDIX B: DETAILED RESPONDENT DEMOGRAPHICS

## Respondents by Country

| | |
|---|---|
| U.S. | 125 |
| U.K. | 104 |
| France | 104 |
| Germany | 100 |
| Japan | 100 |
| Singapore | 100 |
| China | 100 |
| Finland | 52 |

## Respondents by Select Job Titles

| | |
|---|---|
| Chief Technology Officer (CTO) | 58 |
| VP of Engineering | 8 |
| VP of Software Development | 23 |
| Director of Software | 9 |
| Head of Embedded Software | 11 |
| Embedded Software Engineer | 9 |
| Embedded Systems Developer | 14 |
| Senior Firmware Engineer | 8 |
| Product Security Engineer | 69 |
| Embedded Security Engineer | 68 |
| Cybersecurity Engineer | 29 |
| Functional Safety Manager | 20 |
| Functional Safety Engineer | 9 |
| Quality Manager | 19 |
| AI/ML Engineer | 30 |
| Technical AI Product Manager | 41 |

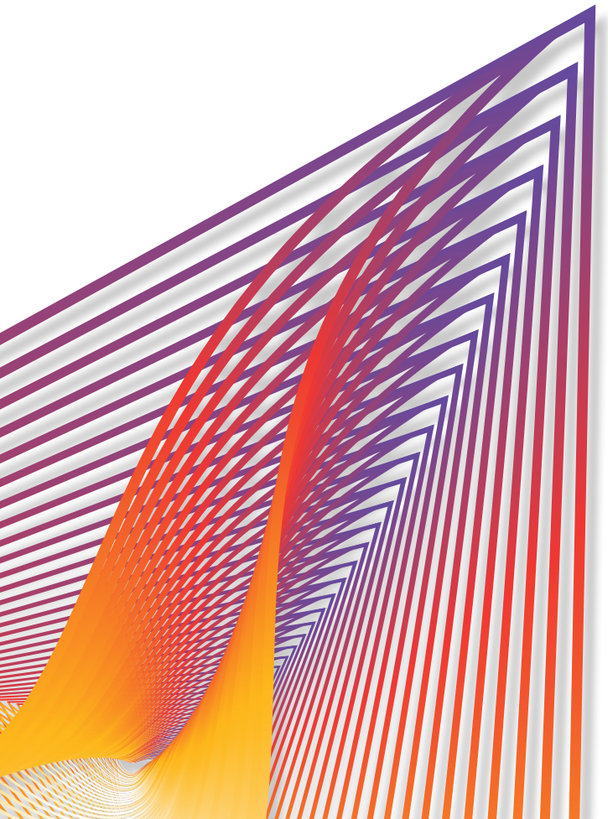
## Respondents by Company Size (Number of Employees)

| | |
|---|---|
| 1–9 | 51 |
| 10–49 | 92 |
| 50–99 | 136 |
| 100–249 | 219 |
| 250–500 | 167 |
| More than 500 | 120 |

## Respondents by Industry Sector

| | |
|---|---|
| Application/Software | 144 |
| Cybersecurity | 132 |
| Technology | 122 |
| Healthcare | 72 |
| Systems Engineer, Functional | 71 |
| Manufacturing | 63 |
| Automotive | 61 |
| MedTech | 44 |
| Government | 43 |
| Transportation | 38 |
| Telecommunication/ISP | 37 |
| Utilities | 26 |

# ABOUT BLACK DUCK



Black Duck® meets the board-level risks of modern software with True Scale Application Security, ensuring uncompromised trust in software for the regulated, AI-powered world. Only Black Duck solutions free organizations from tradeoffs between speed, accuracy, and compliance at scale while eliminating security, regulatory, and licensing risks. Whether in the cloud or on premises, Black Duck is the only choice for securing mission-critical software everywhere code happens. With Black Duck, security leaders can make smarter decisions and unleash business innovation with confidence. Learn more at www.blackduck.com.


BLACK DUCK®