



AI ALONE WON'T MAKE YOU A NEXT-GEN DEVELOPER

AI is capturing attention, but organizations can't lose sight of software engineering fundamentals. Here's why you need both to succeed.



TABLE OF CONTENTS

Welcome to the Elephant in the Room!	03
Introduction: The AI Developer Boom	04
Top Use Cases: AI in Software Development	05
The Challenge: Speed Without Skill	14
The Confidence Gap: Distance Blurs Risk	15
The Insight: Good Developers Make Great AI Users	16
The Solution: Building the Next-Gen Developer	17
Conclusion	20
Glossary	21
Citations	23
About Exact Market	24

Top Use Cases

The Challenge

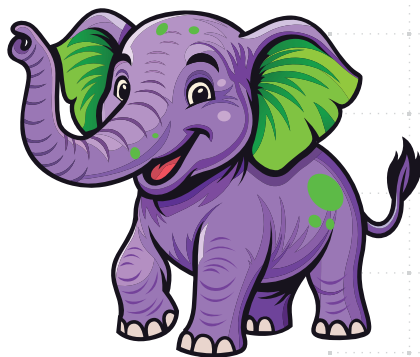
The Confidence Gap

The Insight

The Solution

WELCOME TO THE ELEPHANT IN THE ROOM!

You have in your hands a guide designed to call out the elephant in the room: a topic that's too important to be ignored but isn't getting the attention it deserves.



THE ELEPHANT

AI Alone Won't Make You a Next-Gen Developer

The past two years have been a blur of demos, copilots, and productivity charts. We've all seen the promises: ship twice as fast, deliver twice as much. Maybe that's true for some. But faster doesn't always mean better.

Anyone who's used AI to code knows the thrill of when it nails a fix—and the frustration when that same fix creates a new bug three files away.

AI has completely changed how we build, but it hasn't impacted what good looks like. Code still needs structure. Systems still need security. Developers still need judgment.

Top Use Cases

The Challenge

The Confidence Gap

The Insight

The Solution

INTRODUCTION

The AI Developer Boom

Everyone's Moving Faster

AI has gone from experimental to essential almost overnight. Today, most enterprise development teams use AI-assisted tools somewhere in their workflow, whether it's generating boilerplate code, writing tests, refactoring apps, or documenting APIs.

It's easy to see the appeal:

- Faster delivery cycles
- Smaller backlogs
- Fewer tedious tasks

But the more interesting question isn't about speed. **It's about trust.**

How much of that AI-generated code is reviewed?

How much of it is secure, maintainable, and compliant?

And how much of it do developers truly understand?

Traditional
development process



AI-accelerated
process



Is faster always safer?

Top Use Cases

The Challenge

The Confidence Gap

The Insight

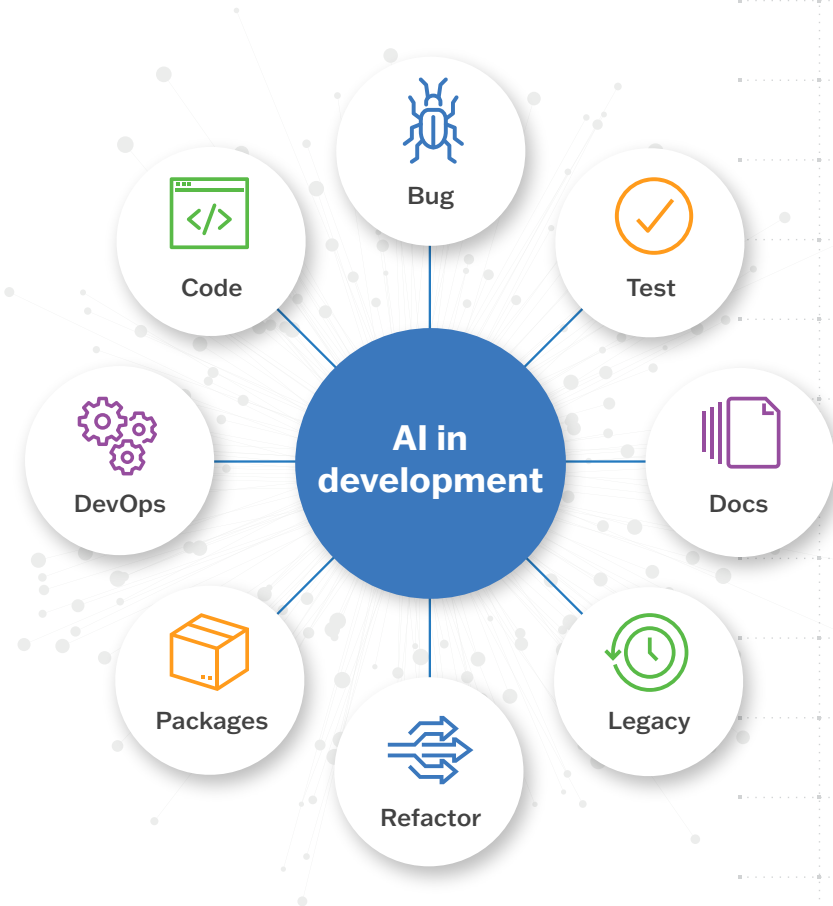
The Solution

TOP USE CASES

AI in Software Development

Enterprises are leveraging AI across the lifecycle in increasingly sophisticated ways.

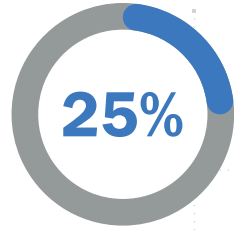
Here's where organizations are seeing dramatic results.



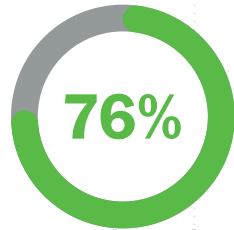
1. Code Generation

In October 2024, Google CEO Sundar Pichai revealed a milestone that would reshape how we think about software development. At one of the world's largest technology companies, employing tens of thousands of engineers working across billions of lines of code, more than a quarter of all new code is now generated by AI.³

Additionally, according to data published in February 2025, Accenture developers showed strong adoption: 81% installed GitHub Copilot on day one of receiving their license, and 67% use it at least five days per week.⁴ The results speak to productivity: Accenture developers retained 88% of GitHub Copilot-generated characters and saw an 84% increase in successful builds.⁵



of Google's new code is AI-generated¹



of developers use or plan to use AI tools²



Top Use Cases

The Challenge

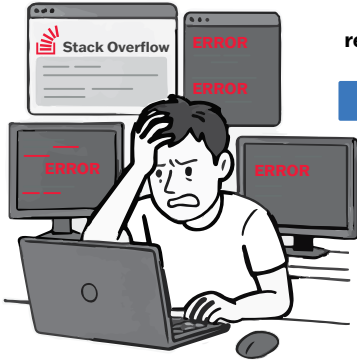
The Confidence Gap

The Insight

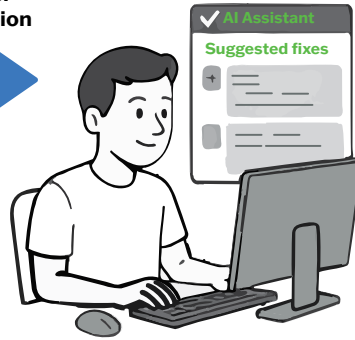
The Solution

2. Debugging and Troubleshooting

TRADITIONAL DEBUGGING



AI-ASSISTED DEBUGGING



AI debugging tools analyze stack traces across millions of code samples, suggest fixes based on similar issues, and explain complex error messages in plain language. The technology is reshaping one of the most frustrating parts of software development.

Amazon's Transformation

\$260M in cost savings.⁶

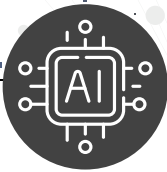
4,500 developer-years saved⁷

By using AI to update and modernize 30,000+ Java applications⁸

In August 2024, Amazon CEO Andy Jassy announced that by using Amazon Q Developer code transformation capabilities, the company had migrated over 30,000 production applications from Java 8 or 11 to Java 17 in just a few months.⁹ This transformation saved an estimated 4,500 developer-years of work and provided \$260 million in annualized efficiency gains from enhanced security and reduced infrastructure costs.¹⁰ Developers shipped 79% of the AI-generated code reviews without any additional changes, demonstrating both speed and quality.¹¹

3. Automated Testing

What once took days of manual work can now be generated in minutes. AI testing tools analyze code to identify untested paths, generate comprehensive test suites, and create edge case scenarios that developers might overlook.



WHAT AI DOES WELL

- Achieve 80% or more coverage quickly
- Generate edge case scenarios
- Create mock objects automatically
- Identify untested code paths



WHAT NEEDS HUMAN REVIEW

- Business logic validation
- User experience testing
- Real-world workflow scenarios
- Performance under load

Top Use Cases

The Challenge





The Confidence Gap

The Insight

The Solution

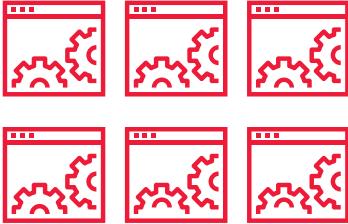
4. Code Documentation

Documentation is the most neglected part of software development. AI is changing this by making comprehensive documentation nearly effortless.

 Inline comments	 API docs
 README files	 Tech specs

Impact on Developer Onboarding

Without AI Docs



6-8 weeks
to full productivity

With AI Docs



3-4 weeks
30-60% faster

AI coding assistants are increasingly used to reduce the time developers spend on routine tasks such as writing code, generating tests, and producing documentation. They also help new developers ramp up more quickly by generating boilerplate, explaining unfamiliar codebases, and providing context-aware suggestions within existing workflows. For smaller teams, these tools can be especially impactful, supporting faster unit test creation and more efficient debugging without adding process overhead.

Top Use Cases

The Challenge

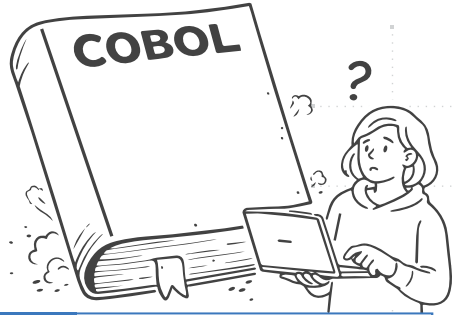
The Confidence Gap

The Insight

The Solution

5. Legacy Code Modernization

Organizations face a critical challenge: billions of lines of mission-critical code running systems that process trillions of dollars, with the developers who wrote them retiring. Generally, traditional modernization takes 5 to 10 years and costs hundreds of millions.



THE COBOL CRISIS

**775-850
BILLION**

lines of COBOL in production globally¹²



70% of global banking transactions run on COBOL¹³

According to a 2024 IBM announcement, there are an estimated 775-850 billion lines of COBOL code globally, with 70% of banking transactions still running on COBOL systems.^{14,15} An estimated 68% of IT executives consider mainframe applications central to their business, and 84% of IBM mainframe customers run COBOL.^{16,17} The shortage of COBOL developers, combined with the critical nature of these systems, makes AI-assisted modernization essential.

IBM watsonx Code Assistant for Z uses a 20-billion-parameter language model trained specifically on COBOL and Java to translate legacy business services into modern, object-oriented code.



FASTER
conversion time



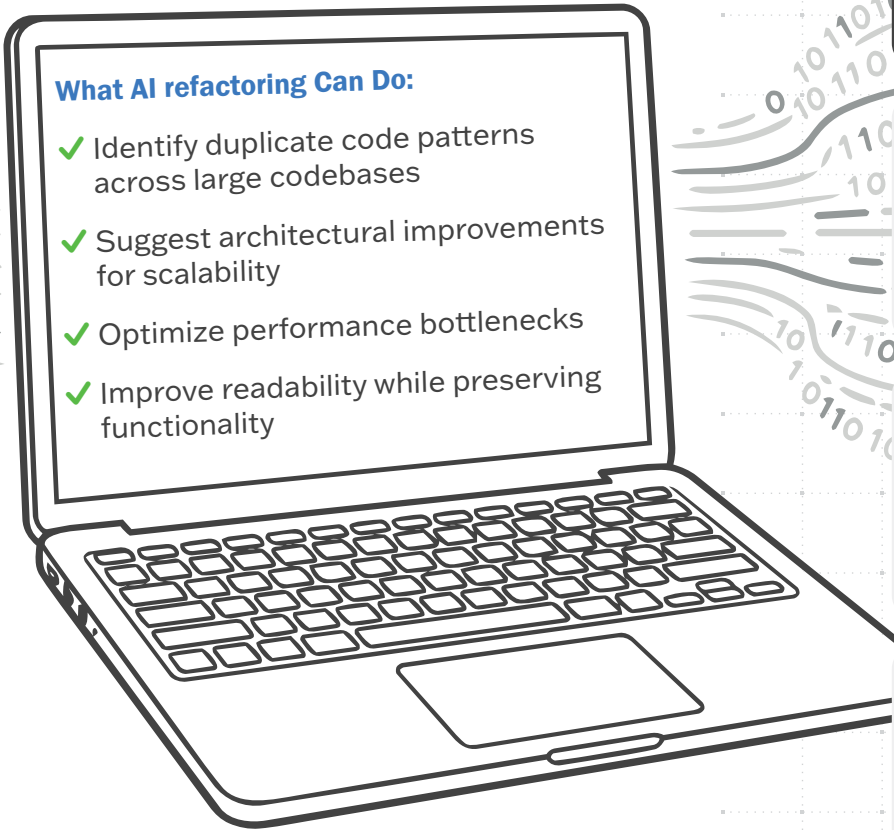
EXPANDED TALENT POOL
Via the move to modern languages

“We are intrigued by the opportunity to leverage generative AI to challenge legacy approaches with material productivity gains.”

Roger Burkhardt, Head of AI, Chief Technology Officer, Broadridge

6. Code Refactoring

As codebases grow, maintaining clean architecture becomes increasingly challenging. AI refactoring tools identify code smells, suggest design patterns, and help teams avoid technical debt.



AI refactoring tools enable organizations to restructure code for better maintainability, which helps speed up review cycles and improve quality metrics. However, these tools work best when developers provide context about business requirements and architectural goals.

Top Use Cases

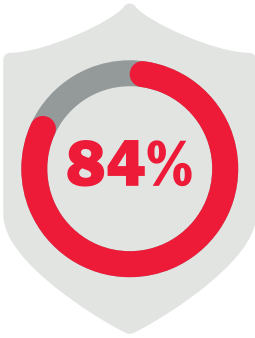
The Challenge

The Confidence Gap

The Insight

The Solution



7. Open-Source Package Discovery



THE OPEN-SOURCE SECURITY CRISIS

84% of codebases contain at least one vulnerability; **74% have high-risk vulnerabilities.**¹⁸

In response, AI tools can help navigate two critical challenges:

 DISCOVERY	 SECURITY
<ul style="list-style-type: none">• Find packages matching requirements• Analyze licensing implications• Check maintenance status• Compare alternatives	<ul style="list-style-type: none">• Identify known vulnerabilities• Flag outdated versions• Suggest secure alternatives• Verify package authenticity

Recent research demonstrates that at least 48% of AI-generated code contains security vulnerabilities.¹⁹ Additional studies show that commercial models reference packages that don't exist in around 5% of their code, while open-source ones do so in nearly 22%.²⁰

From a licensing perspective, AI tools can analyze package licenses and flag potential conflicts, but human review remains essential to ensure compliance with organizational policies.



HALLUCINATION ALERT:

Research found that commercial models reference packages that don't exist in around 5% of their code, while open-source ones do so in nearly 22%.²¹

Top Use Cases

The Challenge

The Confidence Gap

The Insight

The Solution

8. DevOps Automation

AI is streamlining the entire DevOps pipeline, from generating infrastructure as code to setting up monitoring systems. Teams can now generate production-ready CI/CD pipelines in hours instead of days.

AI-GENERATED PIPELINE COMPONENTS



The Human Element



When systems fail, AI can't make the call about rollback vs. hotfix. It can't handle stakeholder communication during incidents. Accountability can't be automated.

AI can generate CI/CD pipeline configurations, infrastructure-as-code templates, monitoring setups, and deployment scripts. However, human oversight remains critical for production systems. Automated pipelines must be reviewed, tested, and validated before deployment.

Top Use Cases

The Challenge

The Confidence Gap

The Insight

The Solution

SPEED WITHOUT SKILL

When “Move Fast” Stops Being Fun

Organizations are constantly looking to “develop faster” and “be more agile” until something breaks in production. AI helps developers code at scale, but without strong engineering discipline, it also multiplies mistakes at scale.

As we’ve discussed, even the best AI models have blind spots. [Let’s review the facts:](#)

SECURITY VULNERABILITIES

At least 48% of AI-generated code contains security vulnerabilities²²

Top issues include:

- Missing input validation
- SQL injection (CWE-89)
- Cross-site scripting
- Hard-coded secrets

LICENSE AND IP RISKS

Commercial models reference packages that don’t exist in around 5% of their code, while open-source ones do so in nearly 22%.²³

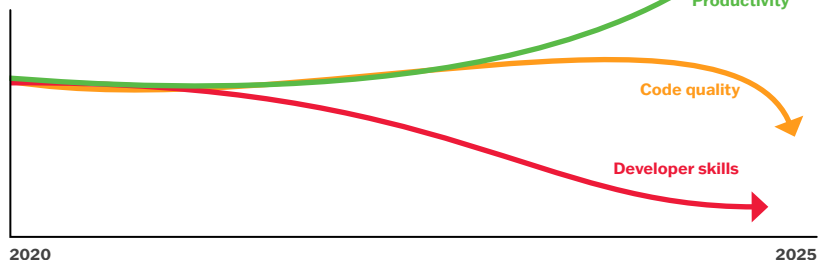
Code pulled from repositories without proper license compliance or attribution creates legal exposure.

TECHNICAL DEBT EXPLOSION

AI-generated pull requests contained approximately 1.7 times more issues overall²⁴

AI tools sometimes deliver code that works today but becomes unmanageable tomorrow. Volume over quality creates mounting debt.

And maybe the most dangerous risk: skill atrophy. The more developers rely on AI for problem-solving, the less they flex their own problem-solving muscles. If we let that trend continue, we’ll end up with engineers who can deploy anything but debug nothing.

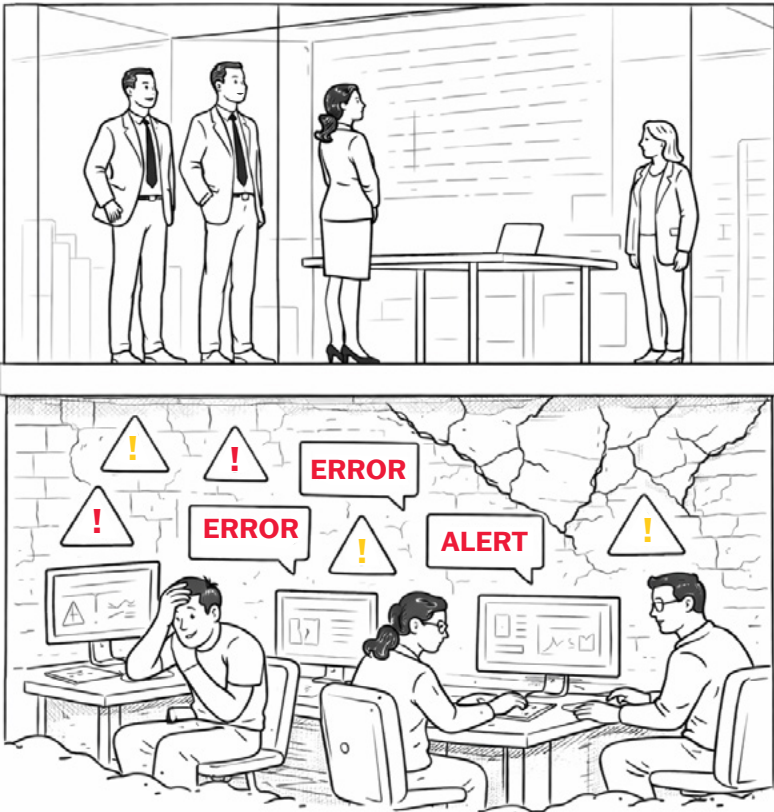


DISTANCE BLURS RISK

Executives Are Ready. Security Teams Are Not.

The further someone is from the code, the safer they think it is. Research from Snyk shows that CTOs and CISOs were two to five times less likely to see security risks from AI-generated code than AppSec practitioners.²⁵ The same study found that while executives rated their organizations as “extremely ready” for AI, developers and security teams disagreed.

This disconnect reveals a dangerous dynamic: as AI accelerates delivery, the visibility of risk gets blurrier the higher up you go. The people closest to the code see the cracks first, but the people furthest away often decide how fast to drive.



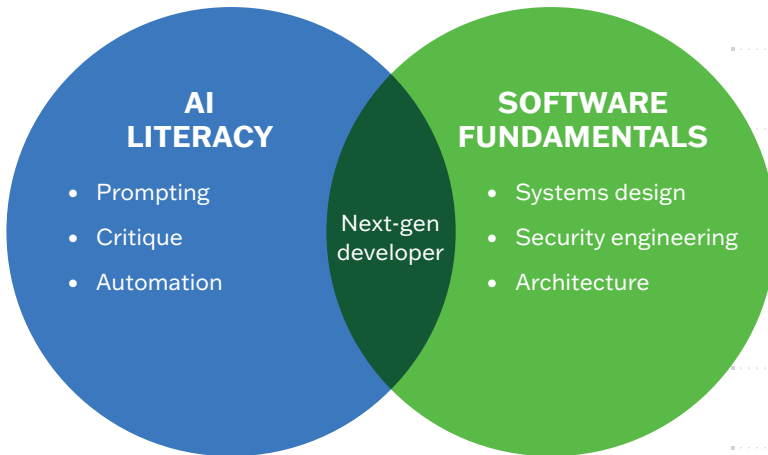
GOOD DEVELOPERS MAKE GREAT AI USERS

Why the Fundamentals Matter More Than Ever

AI can write decent code. But only a developer with deep technical judgment can tell whether that code is right. AI doesn't replace skill. It amplifies it. Vibe coding by a non-technical person may function, but those bugs and flaws become the responsibility of real developers who will spend just as much time fixing the issues.

The best developers in 2026 won't be the ones typing every line. Instead, they will be the ones who combine AI's speed with their own critical eye.

Gartner predicts that by 2027, 75% of hiring processes will include AI proficiency tests—and that through 2026, 50% of companies will start to test for "AI-free" skills that demonstrate the ability to think, debug, and architect without AI assistance.²⁶ That duality defines the next generation of talent.



75%

of hiring processes will include AI proficiency tests by 2027²⁷

Top Use Cases

The Challenge

The Confidence Gap

The Insight

The Solution

THE SOLUTION

Building the Next-Gen Developer

Bending craft, curiosity, and code

The future developer is trained to use AI responsibly.

Their hybrid skill set looks like this:

AI SKILLS

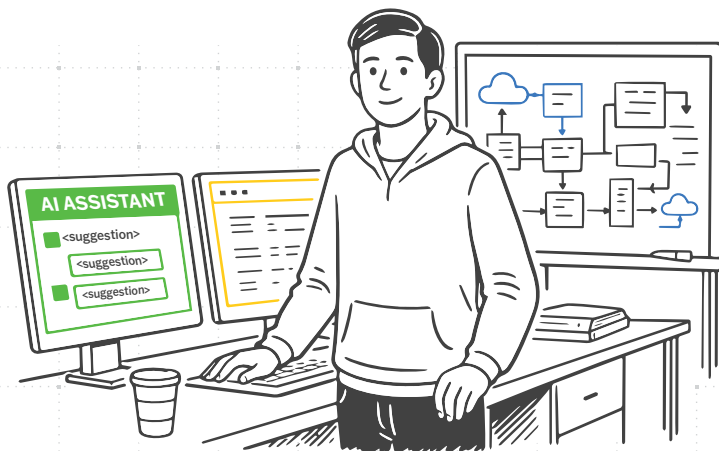
- Prompting effectively
- Reading AI output critically
- Leveraging AI for routine work
- Automating the boring stuff



HUMAN SKILLS

- Systems design
- Secure coding
- Code review
- Collaboration

= Next-Gen Developer



USING AI THE RIGHT WAY ACROSS THE LIFECYCLE



If used wisely, AI fits naturally into every phase of software development.

PHASE 1: PLANNING AND DESIGN

What AI Can Do:

AI excels at brainstorming, generating user stories, and converting requirements into draft specifications. It can quickly create wireframes, suggest database schemas, and identify potential edge cases based on similar projects.

What Humans Must Do:

Define what success looks like. AI can't understand your business goals, user needs, or organizational constraints. Developers must validate technical feasibility, align with architectural standards, and make critical design decisions about scalability, security, and maintainability.

Best Practice:

Use AI to accelerate the ideation phase, but always run design decisions through architecture review. Document why you chose certain approaches over AI suggestions.

Top Use Cases

The Challenge

The Confidence Gap

The Insight

The Solution

PHASE 2: CODING AND REFACTORING

What AI Can Do:	Generate boilerplate code, suggest implementations for common patterns, refactor code for readability, and identify opportunities to reduce duplication. AI can quickly scaffold APIs, database models, and standardize CRUD operations.
What Humans Must Do:	Review every line before committing. Ensure the code follows your team's conventions, integrates properly with existing systems, and handles edge cases. Verify that dependencies are appropriate and security best practices are followed.
Critical Rule:	Require peer review before anything ships. Never merge AI-generated code without human verification. Track which code came from AI to aid future debugging.

Top Use Cases

The Challenge

PHASE 3: TESTING AND QA

What AI Can Do:	Generate comprehensive unit tests, integration tests, and edge case scenarios. AI can achieve high code coverage quickly and identify untested paths. It's particularly good at creating test data and mock objects.
What Humans Must Do:	Catch usability issues, user experience problems, and intent mismatches that AI misses. Test real-world workflows, accessibility, performance under load, and failure scenarios. Verify that tests actually validate business logic, not just syntax.
Quality Gate:	AI-generated tests should be reviewed like production code. Poor tests are worse than no tests because they provide false confidence.

The Confidence Gap

PHASE 4: DEVOPS AND OPERATIONS

What AI Can Do:	Generate deployment scripts, infrastructure-as-code templates, CI/CD pipelines, and monitoring configurations. AI can spot patterns in logs, predict potential failures, and suggest optimizations for resource usage.
What Humans Must Do:	You can't automate accountability. When things break in production, experienced engineers must diagnose root causes, make judgment calls about rollbacks versus hotfixes, and communicate with stakeholders. Humans own the incident response and postmortem process.
Operational Excellence:	Use AI for pattern detection and routine automation but maintain human oversight of production systems. Build runbooks that work without AI so you're not dependent on it during outages.

The Insight

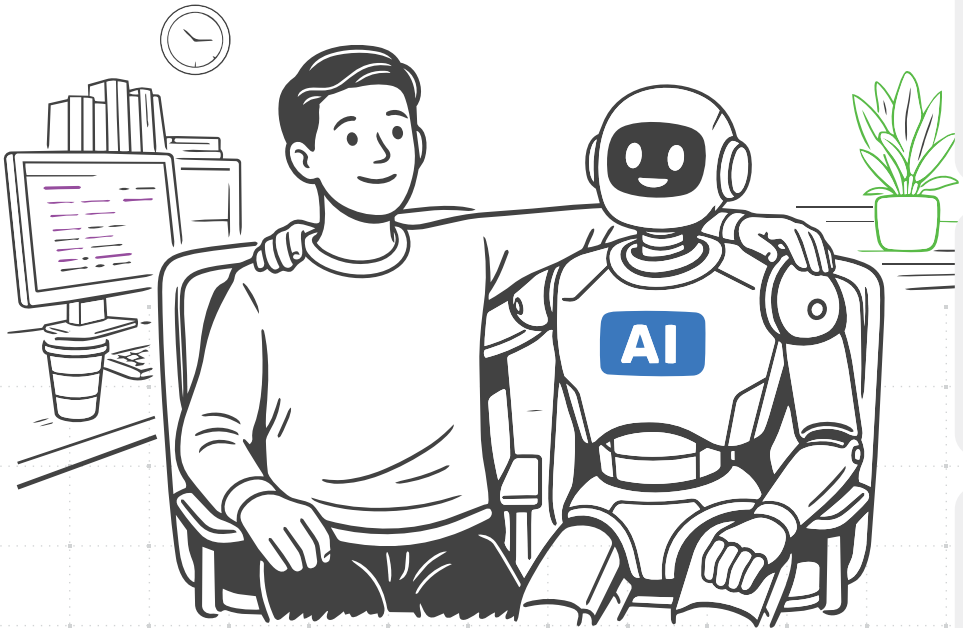
The Solution

CONCLUSION

AI isn't going away. Organizations need to bridge the perception gap with shared accountability between leadership, security, and development. The best teams balance speed with stewardship and innovation. They'll train developers to use AI and to question it—knowing when the answer feels off and how to fix what's broken.

If AI has taught us anything, it's this:

the human element is still the most powerful part of the system.



Top Use Cases

The Challenge

The Confidence Gap

The Insight

The Solution

GLOSSARY

backlog: A collection of outstanding work items that teams have committed to addressing, typically organized by priority. Backlogs often represent pending features, technical improvements, or bug fixes. AI tools may reduce backlog volume by accelerating repetitive or time-consuming development tasks.

continuous integration/continuous deployment (CI/CD): A DevOps practice that automates the process of integrating code changes, running tests, and deploying software. CI/CD pipelines help teams ship updates quickly and reliably. AI can generate much of the configuration, but humans must still verify quality gates and production safety.

common business-oriented language (COBOL): A legacy programming language widely used in banking, finance, and government systems. Billions of lines of COBOL code remain in production, making modernization a major challenge. AI is increasingly used to translate or refactor COBOL systems into modern languages.

code smells: Structural or stylistic indicators that suggest deeper issues within a codebase, such as duplicated logic, overly complex functions, or poor naming. AI refactoring tools can detect code smells and propose improvements, but developers must ensure that changes preserve functional intent.

cross-site scripting (XSS): A security vulnerability where attackers inject malicious scripts into trusted web pages. When users load the compromised page, the script executes in their browser, potentially stealing data or altering site behavior. Poorly reviewed AI-generated code may unintentionally introduce XSS risks.

create, read, update, delete (CRUD): The four fundamental operations performed on persistent data in most applications. AI tools often generate CRUD endpoints, database models, and scaffolding, accelerating initial development while requiring careful human review of security and validation logic.

developer onboarding: The process of integrating new developers into a codebase, team, or organization. Effective onboarding includes context, documentation, foundational knowledge, and mentoring. AI-generated documentation and code explanations can significantly shorten onboarding time when used responsibly.

hallucination: An AI failure mode where the model produces output that is plausible but incorrect, nonexistent, or invented. Hallucinations can introduce design flaws, security issues, or misleading information, making human validation essential for all AI-assisted development.

Top Use Cases

The Challenge

The Confidence Gap

The Insight

The Solution

GLOSSARY

hard-coded secrets: Credentials, API keys, tokens, or passwords embedded directly in source code. This practice creates severe security risks because secrets may be accidentally committed to version control or exposed to unauthorized users. AI-generated code may unintentionally introduce hard-coded secrets unless reviewed.

mainframe modernization: The process of updating or migrating long-standing mainframe applications—often written in languages like COBOL—to modern architectures and frameworks. AI-powered tools can accelerate code translation and refactoring, helping organizations reduce multi-year modernization efforts to months.

package hallucination: A specific type of hallucination where an AI system suggests software packages, libraries, or dependencies that do not actually exist. This poses supply chain risks because attackers may create malicious packages using the hallucinated names.

SQL injection: A critical vulnerability where attackers manipulate input fields to alter backend database queries. If a system does not properly sanitize inputs, injected SQL statements may grant unauthorized data access or manipulation. AI-generated code often omits necessary input validation, increasing SQL injection exposure.

supply chain attack: A cyberattack that targets the software supply chain—such as dependencies, build tools, or distribution channels—rather than the application itself. Package hallucination, outdated libraries, or insecure dependencies can expose organizations to supply chain compromise.

technical debt: The accumulated cost of past shortcuts, rushed decisions, or quick fixes in software development. While technical debt may accelerate immediate delivery, it increases long-term maintenance burden. AI can unintentionally magnify technical debt if teams merge code without thorough human review.

vibe coding: A style of using AI tools to produce application code by describing the intended behavior in natural language, without writing traditional code. While accessible to non-developers, vibe coding can result in hidden issues, requiring experienced developers to validate and refine what AI produces.

Top Use Cases

The Challenge

The Confidence Gap

The Insight

The Solution

CITATIONS

CITATIONS

1. Fortune, "[Over 25% of Google's code is written by AI, Sundar Pichai says](#)," October, 2024.
2. Stack Overflow, "[2024 Developer Survey](#)," 2024.
3. Fortune, "[Over 25% of Google's Code is Written by AI, Sundar Pichai Says](#)", October 2024.
4. GitHub, "[Research: Quantifying GitHub Copilot's Impact in The Enterprise with Accenture](#)," May 2024.
5. Ibid.
6. Digiday, "[How Amazon's Generative AI Tool for Developers Is Saving 4.500 Years of Work, \\$260 Million Annually](#)," August 2024.
7. Ibid.
8. Ibid.
9. Ibid.
10. Ibid.
11. Ibid.
12. The Register, "[IBM Says GenAI Can Convert COBOL Code to Java For You](#)," August 2023.
13. IBM Research, "[Application Modernization with IBM Generative AI](#)," July 2025.
14. The Register, "[IBM Says GenAI Can Convert COBOL Code to Java For You](#)," August 2023.
15. IBM Research, "[Application Modernization with IBM Generative AI](#)," July 2025.
16. IBM, "[IBM to Acquire Application Modernization Capabilities from Advanced](#)," Jan 18, 2024
17. VentureBeat, "[IBM Taps watsonx Generative AI to Help Modernize COBOL on Mainframes](#)," August 2023.
18. Synopsys, "[2024 Open Source Security and Risk Analysis Report](#)," 2024.
19. Center for Security and Emerging Technology (Georgetown), "[Cybersecurity Risks of AI-Generated Code](#)," November 2024.
20. Spracklen, Joseph, et al. "[We Have a Package for You! A Comprehensive Analysis of Package Hallucinations by Code Generating LLMs](#)," 2025.
21. Ibid.
22. Center for Security and Emerging Technology (Georgetown), "[Cybersecurity Risks of AI-Generated Code](#)," November 2024.
23. Spracklen, Joseph, et al. "[We Have a Package for You! A Comprehensive Analysis of Package Hallucinations by Code Generating LLMs](#)," arXiv, 2025
24. CodeRabbit, "[State of AI vs. Human Code Generation Report](#)," December 2025.
25. Snyk, "[Secure Adoption in The GenAI Era](#)," 2025.
26. Gartner, "[Gartner Unveils Top Predictions for IT Organizations and Users in 2026 and Beyond](#)," 2025.
27. Ibid.

Top Use Cases

The Challenge

The Confidence Gap

The Insight

The Solution

ABOUT



Founded in 2007, Exact Market is a woman-owned, WBENC-certified business focused on unifying marketing and technology around a shared vision to help enterprises innovate with confidence.

We bring together strategy, creativity, and data to help our clients stand out and stand for something. Our team of strategists, writers, designers, and technologists understands that the future of storytelling and software shares the same foundation: clarity, authenticity, and human connection.

We don't just help you talk about innovation.

We help you live it.

Top Use Cases

The Challenge

The Confidence Gap

The Insight

The Solution

Find out more @ www.exactmarket.com