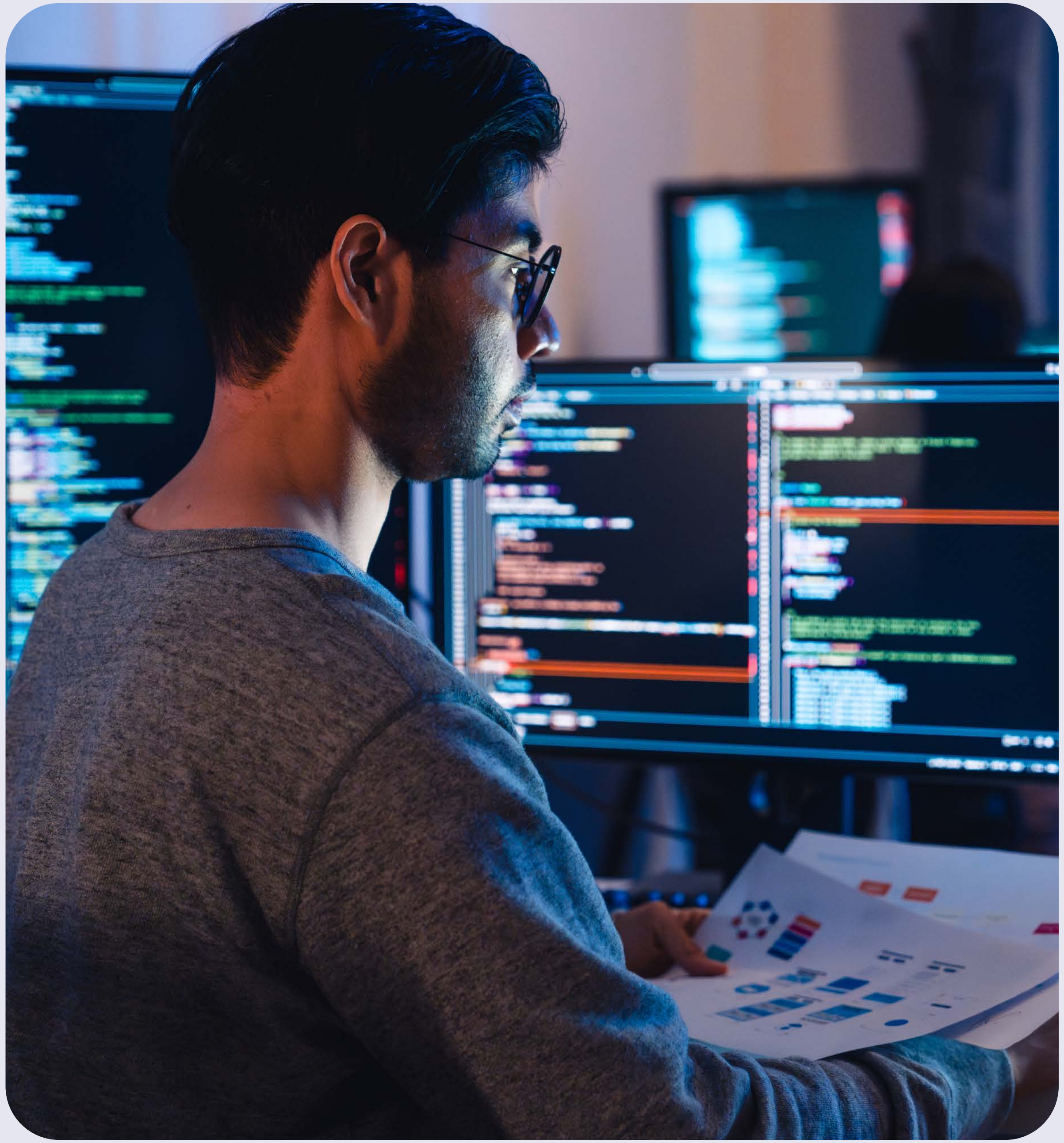
Softtek®

AI-Driven Development:

the new era of code

The new era of code: rewriting the rules of software development

The new era of software rethinks the fundamentals of development



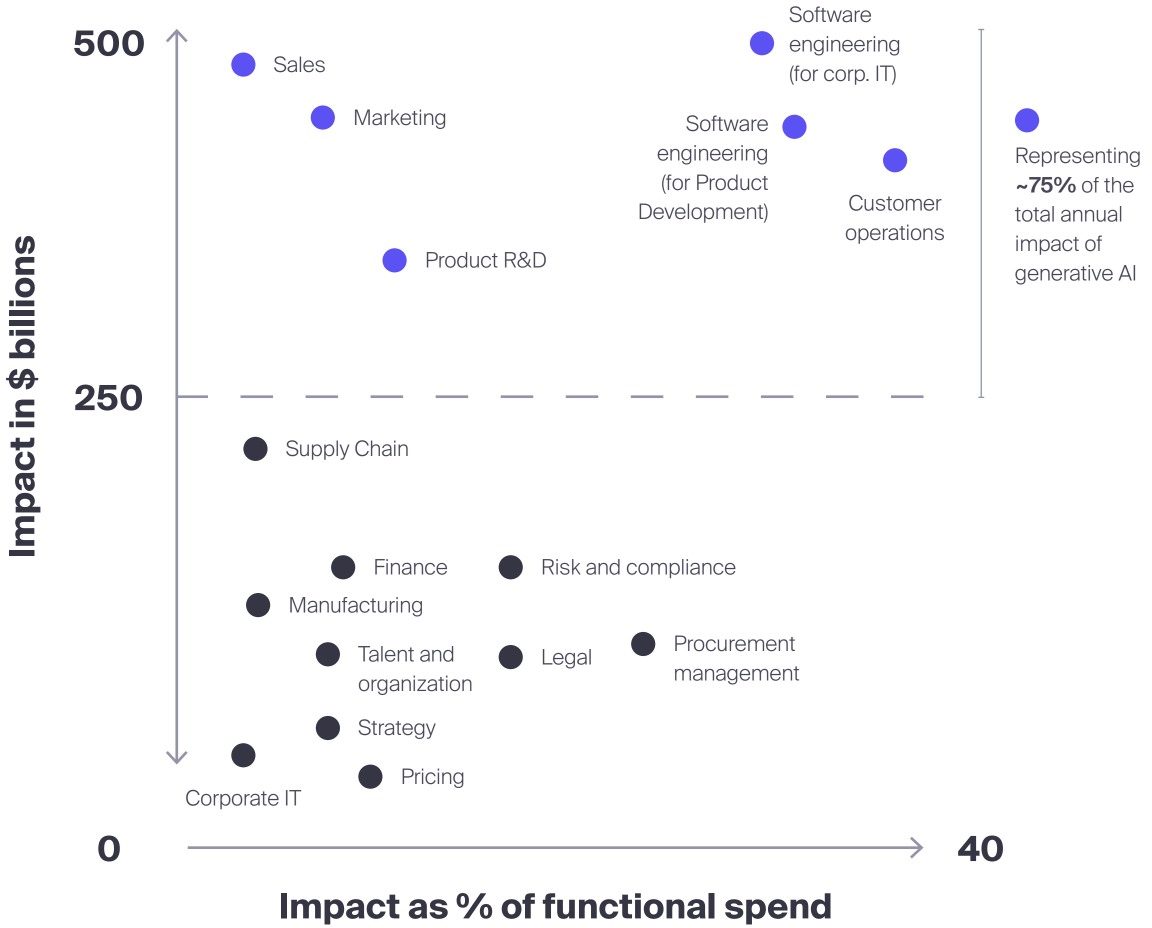
Previously, **software development was largely a manual task.** Developers employed a practical and methodical approach when writing code, relying on their personal experience, collaboration, and tools that provided limited automation. Over time, the industry implemented greater automation, such as continuous integration and continuous deployment (CI/CD) channels, automated testing, and DevOps practices.

However, this human-execution-centric model is now transitioning to a new stage, marked by the emergence of generative artificial intelligence. Unlike previous advancements, these **technologies amplify developer capabilities**, translating requirements expressed in natural language into functional code, generating complete functions, and designing test cases with minimal intervention.

This technological leap **represents a true inflection point.** The new era of code is being shaped by a profound paradigm shift, in which GenAI and other emerging technologies redefine how software is conceived, implemented, and evolved.

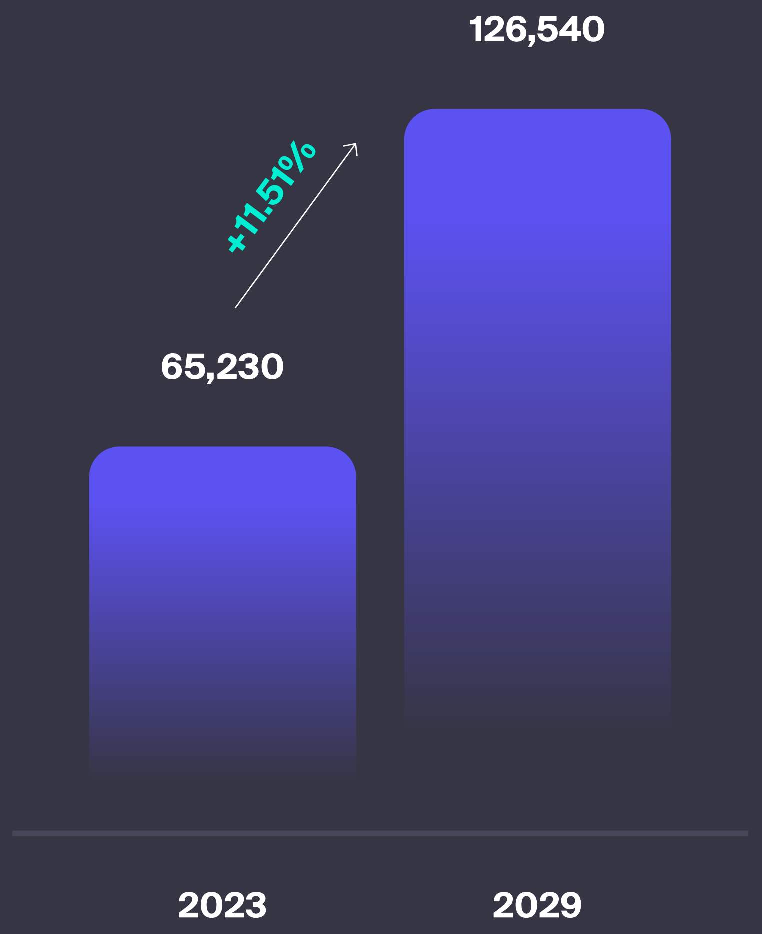
It is estimated that GenAI could contribute between $2.6 trillion and $4.4 trillion annually to the global economy, and a significant portion of this impact will come from software engineering, along with areas like customer operations, marketing and sales, and R&D. In fact, t**he global software engineering market was valued at $65.23 billion in 2023 and is expected to nearly double, reaching $126.54 billion by 2029**, at a compound annual growth rate (CAGR) of 11.51%. This rapid expansion illustrates how deeply AI-based tools and processes are revolutionizing the industry.

Leveraging generative AI in just a few key functions could generate most of its technological impact in corporate use cases



Global software engineering market size

(In $ millions)



The new rules driven by automation and AI radically transform software development and its paradigm

New rules and practices guide development with artificial intelligence

The adoption of AI in software engineering not only improves productivity but **is also modifying the foundations of how applications are conceived, created, tested, and deployed.** The developer's role, project management practices, and quality assurance practices are being redefined under this new logic.

At the core of this transformation **is AI's ability to autonomously generate code.** These tools act as real-time coding assistants, analyzing context and suggesting entire blocks of code. This advancement does not replace the human developer; rather, it frees them from repetitive and low-value tasks, allowing them to focus on higher-level problems, such as complex algorithm design, performance optimization, or critical error resolution.

Another phenomenon is the new code democratization platforms, now AI-enhanced. These **allow non-technical profiles to directly participate in application development,** describing functionalities in natural language, which the platform automatically translates into functional software.

In the DevOps process, AI improves operational stability through proactive supervision and safer deployments. However, all this requires responsible governance, capable of ensuring transparency, avoiding bias, and maintaining human control over automated processes.

Traditional approach	AI-driven approach
Manual coding	**AI-assisted code generation**
Long manual testing cycles	**Automated testing and quality controls**
Isolated development teams	**Cross-functional teams**
Static software requirements	**Adaptive, data-driven updates**

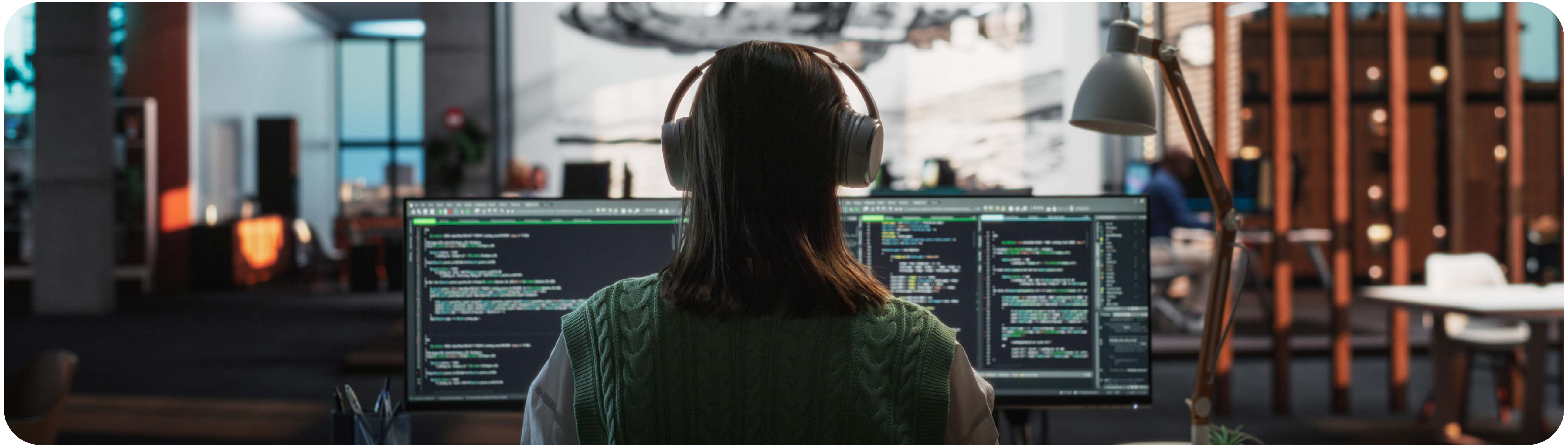
In parallel, **the testing area, one of the classic bottlenecks in the development cycle, has also been transformed by tools that implement artificial intelligence.** These systems generate and execute adaptive test cases, detecting anomalies in software behavior with greater precision and speed than manual techniques. Automating unit and UI tests contributes to maintaining high-quality standards and frees resources previously consumed by tedious tasks.

The influence of AI in testing extends to operations and continuous software delivery. Currently, the industry is actively exploring the **possibility of fully automated software development.** This vision covers the entire cycle: from requirements gathering to implementation and maintenance (Automated Machine Learning).

Speed and accuracy in code generation are not optional, but the key to revolutionizing modern development

The key in modern engineering: agility and accuracy

The environment is constantly evolving and becoming increasingly competitive, fast, and effective. However, **engineering teams still spend a huge amount of time writing code, running tests, and correcting errors.** As applications scale, this process becomes more complex, costly, and difficult to maintain. The lack of automation can lead to longer development cycles, increased technical debt, and higher operational costs. In fact, it is estimated that **developers rewrite, on average, 26% of their code**, which for a medium-sized company can mean more than 4.7 million dollars in annual costs. By contrast, it has been shown that in projects with Proof of Concept (PoC) supported by AI, savings of up to 40% on costs and a 40-60% acceleration in technical decision-making have been achieved.



Therefore, its use becomes a strategic priority:

Accelerating software delivery

By automating processes from code generation to testing and continuous deployments, the time required to launch new functionalities is significantly reduced, allowing shorter delivery cycles and greater responsiveness.

Improving software quality from early stages

Identifying vulnerabilities and anomalous patterns before software enters production (DevSecOps). This increases the robustness of the final product, minimizes the need for post-launch corrections, and reduces the risk of critical incidents.

Reducing development and operations costs

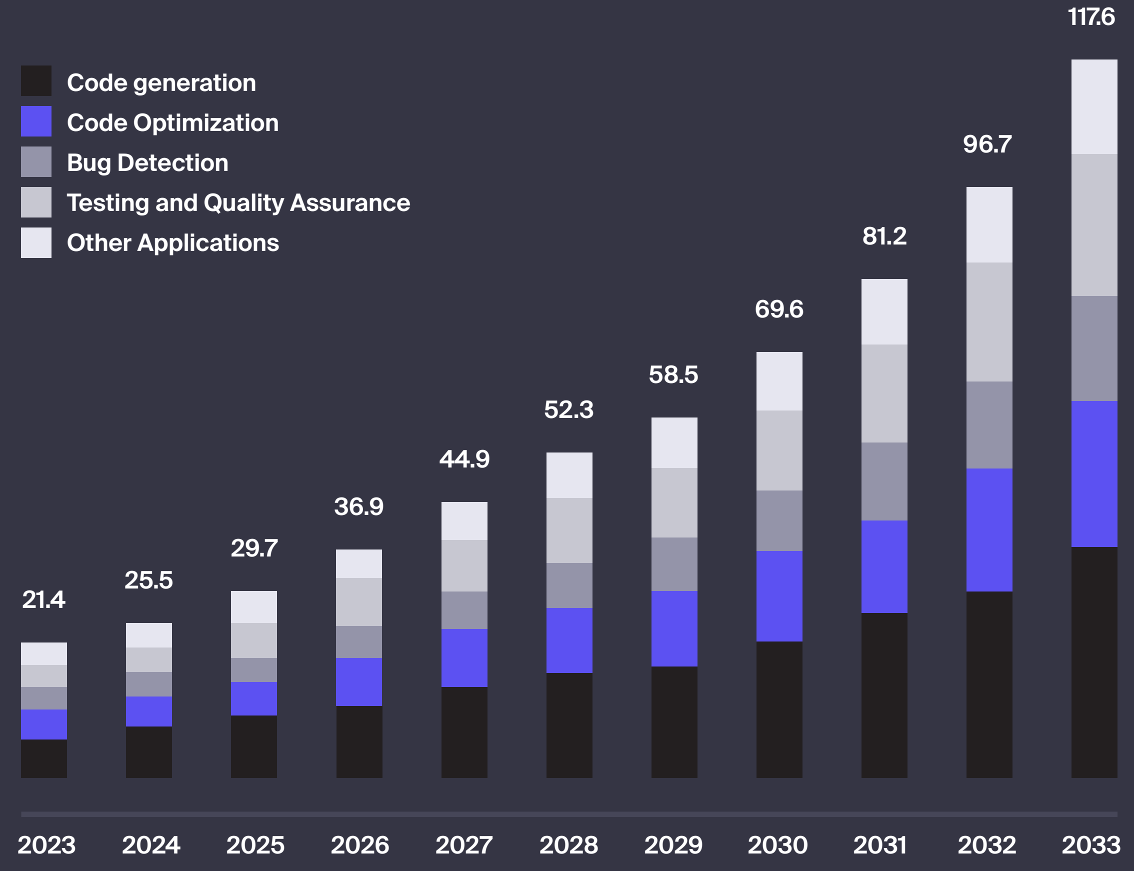
With the support of automated tools, the workload of teams decreases, costly errors are reduced, and infrastructure usage is optimized. This translates into significant operational savings, especially in large-scale projects.

Optimizing human resource allocation

Developers can focus on more strategic and relevant aspects, such as improving user experience, thereby raising team productivity and enabling better talent utilization.

Generative AI market size in software development

(in $ millions)



The market will grow al the CAGR of: **19.1%**

The forecasted market size for 2033 in USD **117.6M**

Not adapting means losing competitiveness, key talent, and accumulating technical debt that compromises future innovation and sustainability

The cost of not adapting: less efficiency, talent loss and technical debt

The automation of internal processes, large-scale personalization, or the creation of cognitive agents are redefining the market, and **companies that do not adapt their development capabilities to this new logic run the risk of being left out of the game.**

The problem is not only technological but also financial, operational, and strategic. While **market leaders build smart platforms and optimize their processes with generative AI**, those sticking to traditional models face an increasing gap in efficiency, costs, and scalability. And this gap continues to grow.

And talent costs are not far behind; it's estimated that **incorporating an AI team today implies highly competitive salaries.** ML engineers in the U.S. can reach up to $180,000 annually, while Senior Data Scientists can reach $200,000 in large corporations. Companies that don't integrate these profiles into their operational architecture will depend on third parties or see their innovation capabilities limited.

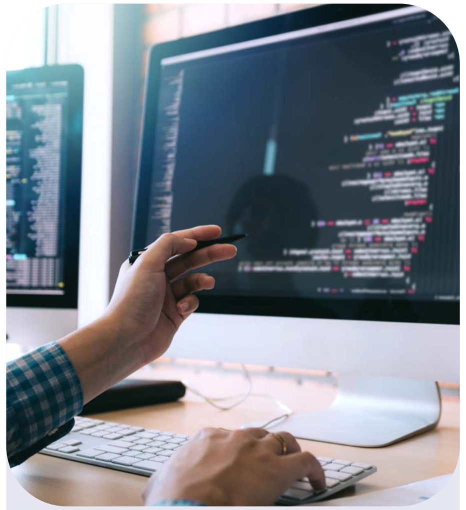
Organizations that don't act now will be paying a price in productivity, relevance, talent, and efficiency that will be increasingly difficult to justify to their stakeholders.

The real cost of not adapting

Hidden costs due to lack of automation

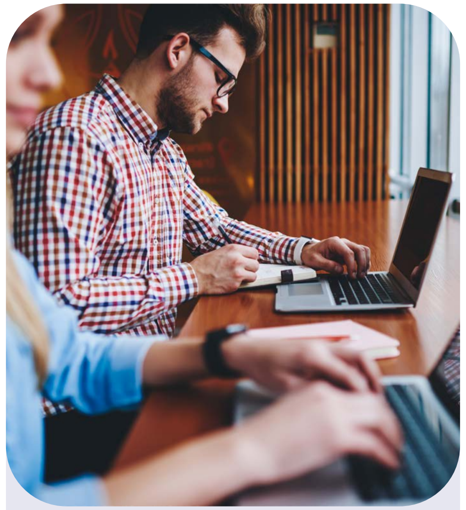
Each non-optimized AI workflow means wasted human time. For example, a single AI model that automates a repetitive workflow can save thousands of hours per year. **Not applying it means recurrently incurring this operational cost.**

Increasing costs in maintenance and evolution



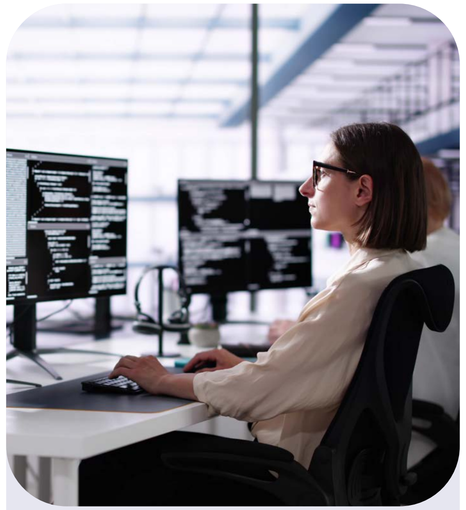
While **traditional software requires 15–25% annual maintenance**, properly designed AI systems allow redistributing that effort through continuous learning and automatic optimization. Without AI, maintenance complexity, costs, and human dependency continue to grow.

Budget misalignment



Maintaining a traditional strategy implies wasting 30–50% of the budget on tasks that AI can partially automate (such as data preparation, testing, or refactoring). Conversely, adopting AI drastically reduces incremental development costs and accelerates return on investment.

Gap in talent and productivity



Developers are migrating to environments that integrate AI as part of the development cycle. Not offering these tools **directly impacts the attraction, retention, and performance** of technical teams.

Dependence on rigid and poorly scalable infrastructures

Training AI models **may require between $10,000 and $100,000 monthly in computing resources depending on the scale.** But not investing in this infrastructure means lacking the capability to evolve smart products or respond to the market in real time.

Augmented engineering redefines development by integrating reliable AI, enhancing decision-making, speed, quality, and adaptability at scale

Foundations of change: AI and augmented engineering

The concept of **Augmented Engineering represents a structural transformation in how we conceive processes and deploy technological solutions.** Integrating artificial intelligence, Machine Learning, augmented reality (AR), and generative tools at the core of practice, enhancing human capabilities to tackle the technical and organizational complexity of the digital age.

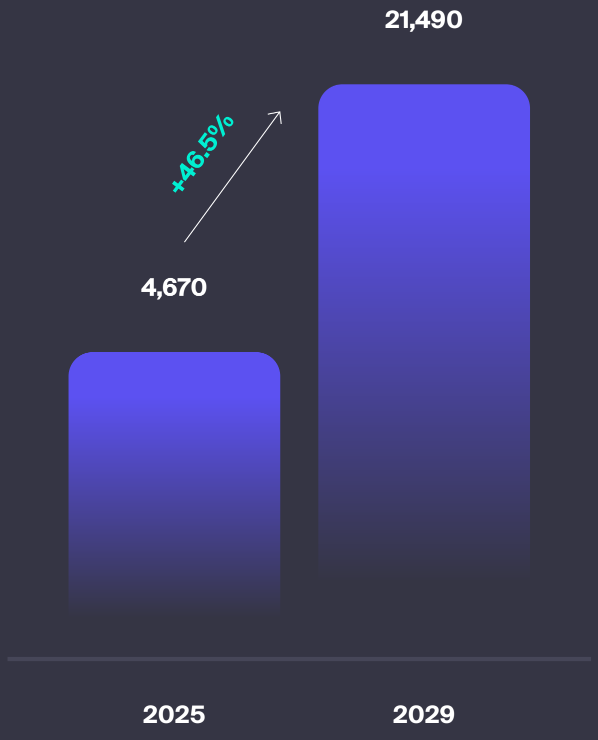
In the field of software development, this evolution is especially significant. The emergence of generative AI has triggered an acceleration in development cycles by **reducing the time required for tasks** that have historically required intense human effort, from writing code to debugging, testing, or deployment. However, this transformation cannot be limited solely to the adoption of coding assistants. The true potential of augmented engineering lies in a comprehensive approach where artificial intelligence acts as a cognitive co-collaborator in all stages of the software lifecycle.

This is based on the symbiosis between the developer's knowledge and the analytical, generative, and predictive capabilities of AI systems. Instead of replacing the professional, AI amplifies human capabilities by correcting errors proactively, suggesting structural improvements, and accelerating automated tests. **It is intelligence that, when well integrated, does not interrupt creative flow but rather enhances it.**

According to a study, **by 2028, 75% of business developers will use AI-based coding assistants**, compared to less than 10% in 2023. Moreover, it's estimated that 80% of software lifecycle activities will involve code generation through GenAI by 2025, increasing productivity by up to 75% in specific use cases.

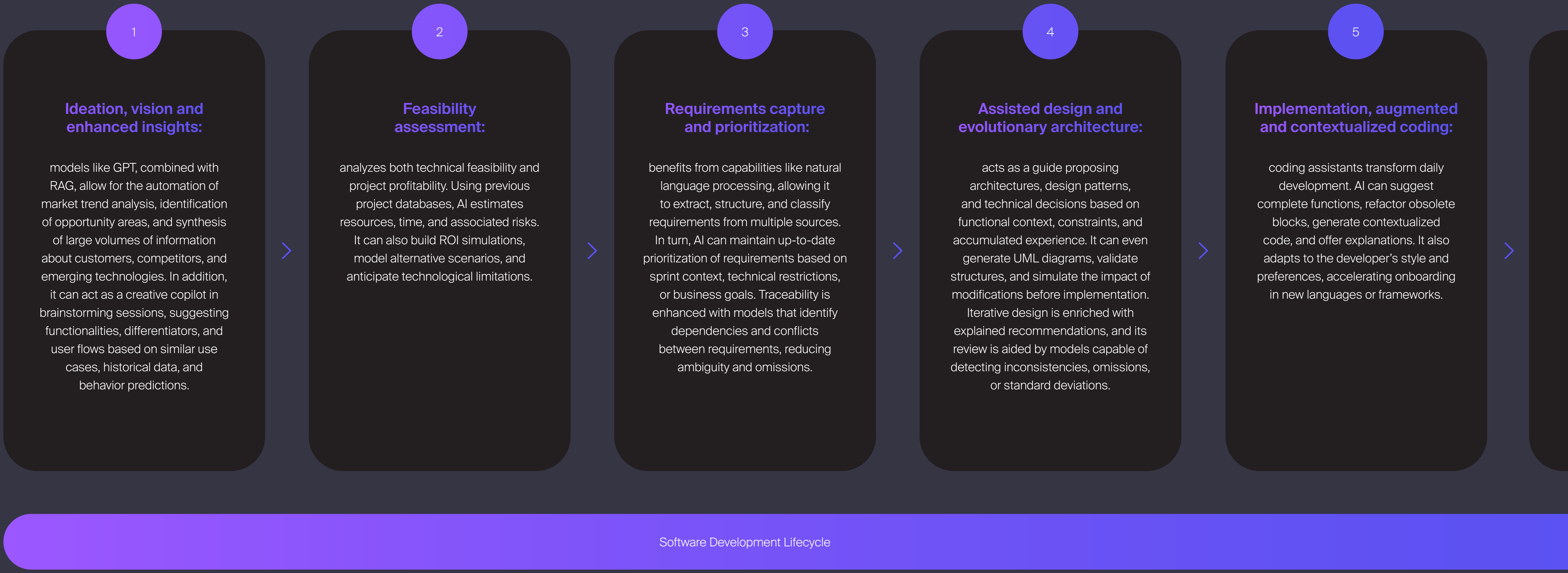
However, the path to augmented engineering is not free of challenges. Unlike other more exploratory or tolerant technological environments, software development demands levels of accuracy, structural validation, and regulatory compliance that generative models cannot fully assume on their own. **This implies a technological and organizational architecture capable of supporting human-machine collaboration**, with intelligent orchestration layers, automation platforms, scalable Cloud environments, observation tools, autonomous security tools, domain-specific models, and continuous feedback mechanisms.

AI-Augmented Software Engineering Market Size (in $ Million)



Augmented engineering not only automates tasks, but also amplifies human capability for decision-making and continuous adaptation in development

Augmented software engineering in the development lifecycle



1

Ideation, vision and enhanced insights:

models like GPT, combined with RAG, allow for the automation of market trend analysis, identification of opportunity areas, and synthesis of large volumes of information about customers, competitors, and emerging technologies. In addition, it can act as a creative copilot in brainstorming sessions, suggesting functionalities, differentiators, and user flows based on similar use cases, historical data, and behavior predictions.

2

Feasibility assessment:

analyzes both technical feasibility and project profitability. Using previous project databases, AI estimates resources, time, and associated risks. It can also build ROI simulations, model alternative scenarios, and anticipate technological limitations.

3

Requirements capture and prioritization:

benefits from capabilities like natural language processing, allowing it to extract, structure, and classify requirements from multiple sources. In turn, AI can maintain up-to-date prioritization of requirements based on sprint context, technical restrictions, or business goals. Traceability is enhanced with models that identify dependencies and conflicts between requirements, reducing ambiguity and omissions.

4

Assisted design and evolutionary architecture:

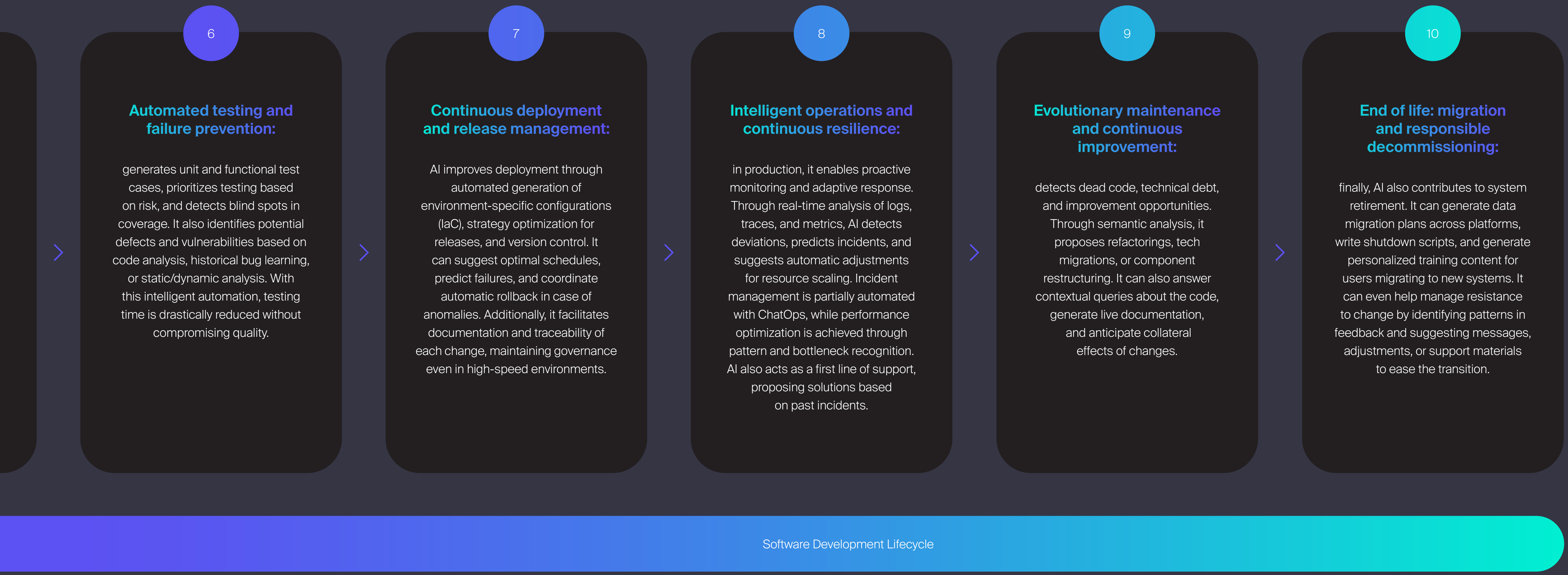
acts as a guide proposing architectures, design patterns, and technical decisions based on functional context, constraints, and accumulated experience. It can even generate UML diagrams, validate structures, and simulate the impact of modifications before implementation. Iterative design is enriched with explained recommendations, and its review is aided by models capable of detecting inconsistencies, omissions, or standard deviations.

5

Implementation, augmented and contextualized coding:

coding assistants transform daily development. AI can suggest complete functions, refactor obsolete blocks, generate contextualized code, and offer explanations. It also adapts to the developer's style and preferences, accelerating onboarding in new languages or frameworks.

Software Development Lifecycle



6

Automated testing and failure prevention:

generates unit and functional test cases, prioritizes testing based on risk, and detects blind spots in coverage. It also identifies potential defects and vulnerabilities based on code analysis, historical bug learning, or static/dynamic analysis. With this intelligent automation, testing time is drastically reduced without compromising quality.

7

Continuous deployment and release management:

AI improves deployment through automated generation of environment-specific configurations (IaC), strategy optimization for releases, and version control. It can suggest optimal schedules, predict failures, and coordinate automatic rollback in case of anomalies. Additionally, it facilitates documentation and traceability of each change, maintaining governance even in high-speed environments.

8

Intelligent operations and continuous resilience:

in production, it enables proactive monitoring and adaptive response. Through real-time analysis of logs, traces, and metrics, AI detects deviations, predicts incidents, and suggests automatic adjustments for resource scaling. Incident management is partially automated with ChatOps, while performance optimization is achieved through pattern and bottleneck recognition. AI also acts as a first line of support, proposing solutions based on past incidents.

9

Evolutionary maintenance and continuous improvement:

detects dead code, technical debt, and improvement opportunities. Through semantic analysis, it proposes refactorings, tech migrations, or component restructuring. It can also answer contextual queries about the code, generate live documentation, and anticipate collateral effects of changes.

10

End of life: migration and responsible decommissioning:

finally, AI also contributes to system retirement. It can generate data migration plans across platforms, write shutdown scripts, and generate personalized training content for users migrating to new systems. It can even help manage resistance to change by identifying patterns in feedback and suggesting messages, adjustments, or support materials to ease the transition.

Augmented engineering requires AI with systemic vision, organizational adaptability, and alignment between people, processes, and technology

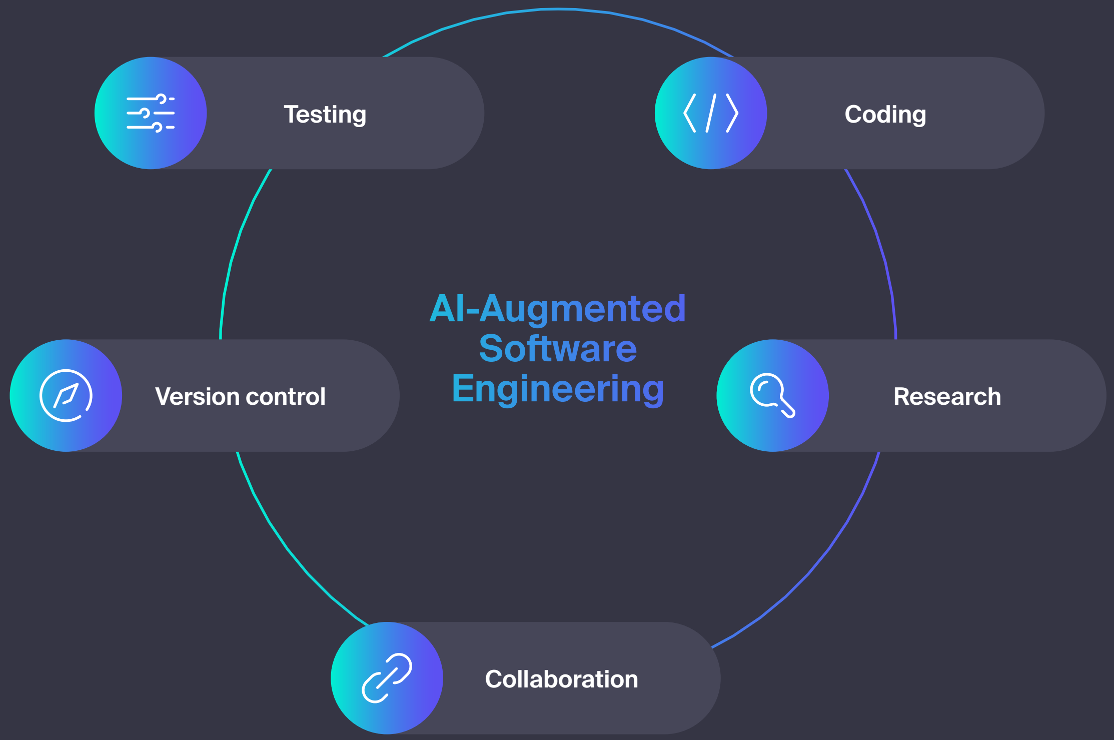
AI-Augmented Software Engineering (AIASE)

However, in critical engineering environments, such as financial, healthcare, or public administration, **the challenges are greater.**

For AI to generate real value in these environments, it must be endowed with extended capabilities. First, systems must be able to process complex inputs, such as technical schemes, simulations, business flows, or legal requirements, **to generate software that works and respects the domain's structural, regulatory, or operational constraints.** This is joined by automatic compliance validation, incorporating sectoral rules, regulatory standards, and audit requirements from the design phase—a compliance by design approach.

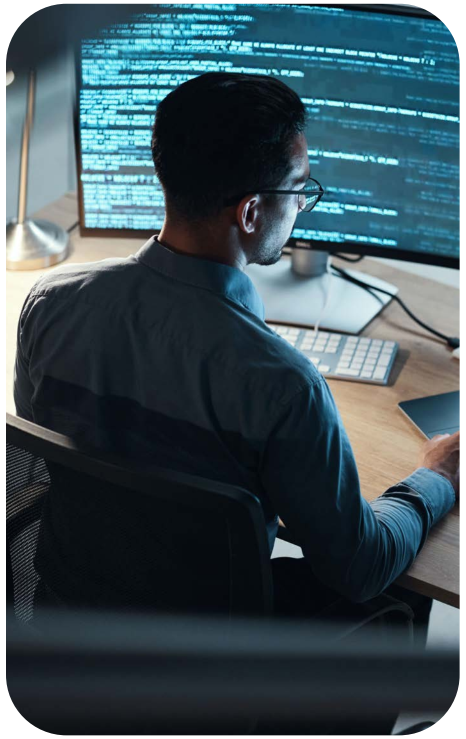
In turn, ensuring the traceability of each algorithmic decision is equally essential—from the data source to the impact of any change in production. This is critical in regulatory frameworks such as the EU AI Act or sector-specific regulations like financial (Basel III, EBA guidelines) or healthcare (HIPAA, MDR). And even as automation grows, in these environments **significant human control (human-in-the-loop)** remains mandatory, where systems can assist but critical decisions must still be made by a qualified expert.

AIASE provides constant support across various areas of software development



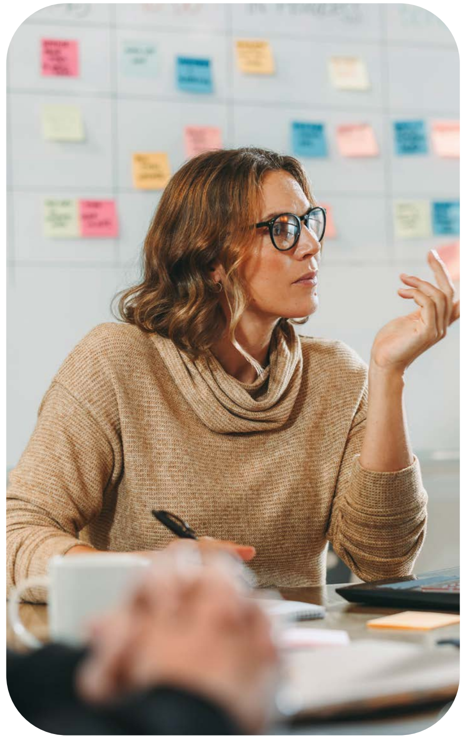
Its impact can be clearly observed when broken down into five key areas:

Coding



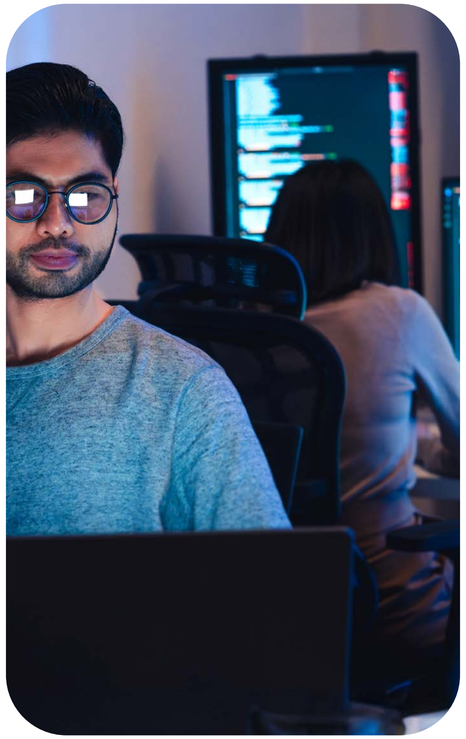
AI enables assisted code generation and review, based on NLP and project context.

Research



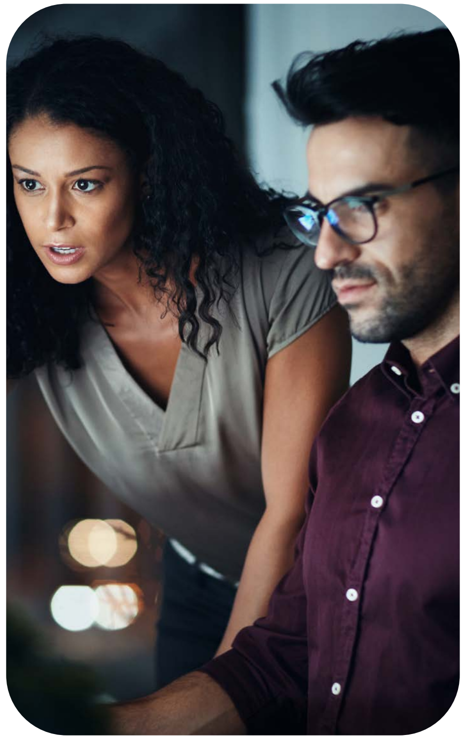
AI-augmented IDEs act as knowledge hubs, facilitating uninterrupted technical information and documentation.

Version control



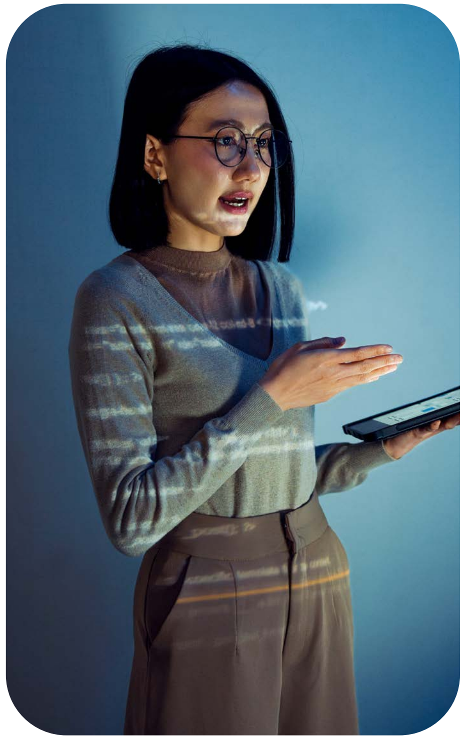
through continuous analysis, detects regressions, anomalies, bottlenecks, and technical debt in real time.

Collaboration



these solutions notify, document, and share context automatically, maintaining team coherence.

Testing



AI can generate, prioritize, and maintain intelligent and adaptive tests, including self-healing cases in response to changes.

There are ethical, technical, and cultural challenges that require a clear strategy for governance, scalability, and responsible adoption

Strategic challenges of AI-enabled software development

High initial investment:	Privacy and data security risks:	Shortage of specialized talent:	Integration with legacy systems:	Explainability and trust:	Ethical risks and algorithmic bias:	Loss of the human factor:
AI solutions may involve significant costs for licensing, infrastructure, and specialized talent. This represents a particularly critical entry barrier for small organizations or those with legacy environments.	AI models require large volumes of data, which often include confidential information, raising concerns about regulatory compliance (e.g., GDPR, CCPA) and the protection of sensitive data.	Effective AI adoption requires knowledge in Machine Learning, data engineering, MLOps, and specific tools—profiles that are scarce and in high demand.	Legacy systems may present technical barriers to the integration of modern AI solutions, which slows down adoption and generates operational frictions.	The decisions generated by AI models are often not interpretable by technical teams or stakeholders, which hinders validation, compliance, and cultural adoption.	AI may amplify existing biases in data or poorly trained models, resulting in unfair or discriminatory outcomes. Furthermore, excessive automation may generate labor tensions and resistance to change.	Intensive use of AI may lead to technological dependency, loss of creative thinking, or erosion of soft skills such as empathy, contextual judgment, or user-centered design.
			Recommendation			
• Solutions with low entry cost • Cloud subscription-based tools • Locally optimized open source models	• Tools with end-to-end encryption, access control and auditing • Strong governance and monitoring • Hybrid architectures that keep data locally	• Upskilling through training • Collaborate with expert consulting firms • Establish internal communities of AI practice	• Progressive integration strategies (phased approach) • Solutions that operate on surface layers without modifying the core legacy • Design APIs that connect legacy systems with intelligent capabilities.	• AI solutions that incorporate XAI mechanisms • Traceability and reasoning visualizations • Documentation of system decisions for all users	• Clear ethical frameworks to govern its use • Regular audits to assess biases and review its social impact • Design AI as a complementary technology	• Human-machine collaboration tools • Promote collaborative models • Evaluate the impact on the team's overall performance

Among these challenges, latency also emerges as a critical barrier for AI-powered systems in production

Latency, the new “downtime” of AI

In the context of generative applications, latency has emerged as a disruptive factor just as service outages have historically been: it is the new “downtime”. The underlying premise is clear: **if a response generated by AI doesn’t arrive on time, it doesn’t arrive with value.** With the rise of conversational interfaces and autonomous agents, response time is expected to resemble that of a human conversation. This challenge is even greater with reasoning models, which may offer higher-quality answers if given additional time to “think.”

In software development, **latency is the time that passes between a developer requesting an action and the system’s AI responding.** This delay, measured in seconds, may seem trivial in other contexts, but in a creative flow it represents a real interruption.

The effect of latency is direct—an AI that responds slowly loses value as it discourages use, disrupts workflow, and creates a perceived experience of being clumsy or unreliable. **This phenomenon is amplified in collaborative contexts.** Delays impact code review efficiency, the quality of agile interactions, and team morale.

It also introduces a new kind of technical debt. As teams increasingly rely on AI for routine or critical tasks, lack of immediacy degrades the overall experience, slows long-term adoption, and may eventually compromise operational effectiveness. Quantitatively, **it is estimated that in optimal scenarios, low-latency AI can improve productivity by over 20%**, but if the tool itself responds with constant delays, those benefits are eroded until they become counterproductive.

As organizations integrate generative AI systems into their development pipelines, latency becomes a new KPI. It’s no longer enough for the model to be accurate or versatile; it must also be fast, stable, and consistent. Leading companies are beginning to redesign their architectures to meet this challenge, using strategies such as:

Edge inference | where models are partially run on local devices to reduce round-trip time to the cloud.

Lightweight adapted models | Such as LLaMA 3, Mistral or other versions optimized for fast inference.

Contextual caching | reducing the need to process from scratch in every interaction.

Optimized RAG architectures | combining context retrieval with localized generation.

An organization that succeeds in minimizing the latency of its augmented tools gains a strategic position to lead in speed, quality, and adaptability. Conversely, those that tolerate slow, unresponsive, or frustrating tools risk falling behind in the digital transformation race.

Augmented software engineering drives a new paradigm with intelligent, adaptable, fast, and human code

Future vision, the new DNA of software development

In the current decade and the ones to come, a broad catalog of emerging technologies will reach widespread adoption and radically transform the way organizations design, develop, and operate software. **This will make it possible to redirect talent toward high-impact areas such as strategic initiatives, resolution of technical uncertainty, or co-creation with the business,** while at the same time improving operational resilience. Testing environments will be enriched with non-obvious code paths, capable of detecting failures before they escalate, and offering automated solutions. This will lead to lower technical debt, greater robustness, and continuous delivery aligned with business objectives.

From manual coding to human-AI collaboration

Developers no longer work alone—thanks to coding assistants, **cognitive load is reduced and delivery is accelerated.**

87%

of developers said GitHub Copilot helped them **preserve mental energy during repetitive tasks.**

Adaptive codebases

Repositories are no longer static; they learn and evolve. Models trained on historical **tests** and deployments can **anticipate errors, suggest refactorings, or apply security patches.**

30%

of the **code suggestions from GitHub Copilot were accepted** on average by users.

Democratization of development

AIASE promotes inclusive access to software development. LCNC platforms with generative capabilities allow **users without technical backgrounds to contribute to the creation of digital products.**

75%

of large corporations will use at least four LCNC tools for both **IT applications and citizen development initiatives.**

New roles and skills

Developers are taking on the role of intelligent systems designers. A key competency will be **knowing how to collaborate with autonomous systems, validate outputs, understand their logic, and make ethical and responsible decisions.**

67%

of **mature AI organizations are creating new roles related to AI,** with 87% already having AI teams.

More quality, fewer errors

Various tools are reducing thousands of production errors **through testing and automated refactoring,** thanks to augmented software engineering.

87%

fewer production defects and 70% **less time in manual reviews thanks** to AI-assisted code review tools.

Key Trends in AI-Driven Development

The new software economy is built with assisted code, LCNC, VibeOps, autonomous agents, and intelligent automation

$5.000B
Global IT spending in 2024

$850.000B
Software development market size by 2028

Key trends toward the future of software development

As organizations face increasingly restrictive economic environments, software development not only persists but continues to reinvent itself. Investments in digital transformation keep growing, with estimates indicating that **global IT spending reached $5 trillion in 2024**, with a strong focus on agile platforms, artificial intelligence, and productivity-boosting tools.

In addition, the software development market is expected to exceed $850 billion by 2028. This demonstrates that **software continues to be a driver of digital transformation.** This context is propelling a new era of code, marked by acceleration, human-AI collaboration, and sustainability.

This outlook demands a rethinking of how software is built, with what tools, by whom, and with what impact. In this scenario, **a series of key trends are emerging that are redefining the pillars of development**: from the massive incorporation of artificial intelligence to the democratization of code, digital sustainability, new developer roles, and the evolution toward platforms as an operating model.

What was once a sequential, handcrafted, and highly specialized process is now becoming a collaborative practice—AI-assisted, more open, faster, and aligned with business objectives. Understanding these trends is not just a matter of innovation, **it is a roadmap to ensure future competitiveness.**

Key Trends

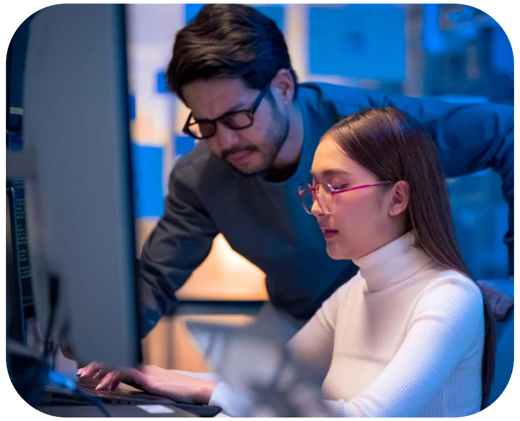
1. Low-Code / No-Code
2. Advanced code assistants
3. Architecture automation
4. Autonomous maintenance agents
5. Reverse engineering and refactoring
6. Vibe Coding & VibeOps

1 Low-Code / No-Code: the other silent driver of change

Software development is no longer reserved exclusively for profiles with years of coding experience. Organizations that once required large technical teams to build even simple applications are now **transforming their processes through Low-Code and No-Code** (LCNC) platforms. These tools allow applications to be created through visual interfaces and predefined components, minimizing the need to write code and increasing the speed, accessibility, and scalability of development.

The incorporation of artificial intelligence into these platforms further enhances their capabilities—simply by describing a functionality in natural language, the system can translate it into an operational application. Not only does this accelerate solution delivery, but it also **further expands the user base that can participate in development,** from business analysts to **Product Managers**, who can now create or adapt tools without constantly depending on an engineering team.

Both SMEs and large corporations can benefit from these competitive advantages, as **48% of IT leaders and C-level executives report having accelerated application development thanks to Low-Code**, while 45% highlight its direct impact on cost reduction. It's no surprise, then, that the global Low-Code platform market was valued at $24.83 billion in 2023 and is projected to grow at an annual rate of 22.5% through 2030.



Democratization of development: when everyone can create software

This movement is based on a digital philosophy **oriented toward the democratization of software development**, removing the barrier of deep technical knowledge. It includes both advantages and new challenges:

Advantages	Challenges
Ease of use and application	Customization restrictions
Immediate visualization of results	Dependency on vendor environments
Significant reduction in development costs	Loss of best design and programming practices due to lack of experience
Lower maintenance burden and higher productivity	Poorly scalable or low-quality solutions without proper governance

80% of companies state that Low-Code improves productivity

79% state that Low-Code reduces operating costs

73% state that Low-Code improves time-to-market

2 Advanced code assistants

The market for AI-assisted development tools reached $12.56 billion in 2024 and will continue growing annually at **24.5% through 2030.**

AI-powered code assistants represent a new layer of intelligence within the development environment. **Their logic is built on large language models**, like GPT or Codex, trained on millions of lines of code and syntactic patterns from multiple languages, **frameworks**, and public repositories. This foundation not only enables code generation, but also an understanding of the functional context of what is being programmed, helping to propose complete solutions and detect problems before they occur during execution.

In this way, **AI acts as a cognitive accelerator that frees up time and mental effort**, allowing people to focus on higher-level conceptual tasks. By reducing friction in repetitive code writing, unit test creation, or structured refactoring, AI assistants allow technical talent to shift more toward design, strategy, and architecture.

The key to their performance lies in their ability to interact seamlessly with the most widely used integrated development environments, like Visual Studio Code, IntelliJ IDEA, or cloud platforms such as GitHub. This means the developer doesn't need to learn a new system, since AI is embedded directly into their usual workflow, becoming a natural extension of their creative process.

More than 30% of new code in companies like Google or Microsoft is already generated by AI, with projections pointing to over 90% by the end of the decade.

The true power of these assistants lies in their ability to understand the broader context in which they operate. That is, they don't just operate on the specific line of code being written—they learn from the project structure, respect internal conventions, and integrate with other ecosystem tools, ensuring consistency and productivity throughout the entire software lifecycle.

Adoption and usage frequency of AI code assistants by developers

49%
use them daily

39%
use them weekly

58%
started using them in the last 6 months

Main concerns of developers about the use of in software engineering

55%
are concerned about the quality of code generated by AI

48%
have concerns about security and privacy

46%
fear reduced demand for developers

3 Architecture automation

The incorporation of artificial intelligence into architecture implies a reconfiguration of the decision-making process, where machine learning models, predictive analytics, and automation **allow systems to be designed more efficiently, scale dynamically, be maintained and documented in real time,** and anticipate failures, bottlenecks, and security risks before deployment.

In parallel, **the automation of architectural documentation represents another leap in value.** Through NLP, systems are capable of extracting technical information directly from specifications, code, or artifacts generated in the development environment and transforming it into structured, readable, and dynamic documentation. This greatly facilitates knowledge transfer, onboarding of new team members, and technical audits in regulated environments.

In addition to design and documentation, AI has also begun to modify deployment and maintenance dynamics, especially through the automation of infrastructure provisioning, integration of intelligent CI/CD pipelines, and continuous system performance monitoring. This gives rise to a new concept of architecture: self-healing architecture, capable of detecting anomalies, diagnosing faults, and applying corrective measures without direct human intervention. Likewise, **self-optimizing systems automatically adjust resources, balance loads, and scale services based on usage patterns, thus achieving a more efficient and cost-sustainable architecture.**

In terms of change management, this means that organizations must begin to integrate AI not as an external tool, but as an internal system capability, embedded from the initial design. However, **this new approach also entails responsibilities.** The introduction of AI into architecture requires reviewing governance practices, incorporating continuous model validation mechanisms, ensuring transparency of recommendations, and maintaining effective human oversight to guarantee alignment with business objectives, system security, and ethical principles.

Modeling automation

Tools like Structurizr or ArchiMate with AI capabilities allow:

- Generate architecture diagrams from requirements.
- Draw dependency maps.
- Propose architectural patterns based on past success cases.
- Reduce manual effort and improve standardization.

Pattern recognition

AI analyzes previous successful architectures to:

- Detect replicable patterns.
- Suggest structures for microservices, caching, message queues, among others
- Optimize performance and scalability from the initial design.

4 Autonomous maintenance agents

Unlike traditional assistants, and driven by automation and pattern detection, **autonomous assistants are capable of breaking down objectives, planning tasks, writing and testing code, and learning from the outcome without human intervention.**

One of the most significant advances is the shift from t**raditional corrective maintenance toward a predictive and preventive model**, thanks to models that analyze historical failure patterns, system usage, and performance. These tools can anticipate incidents before they occur, enabling actions that reduce downtime, optimize resource usage, and prevent service interruptions.

In parallel, the ability to analyze large volumes of logs, metrics, and telemetry in real time allows systems to **identify anomalies, classify errors automatically, and suggest specific fixes.** These processes are becoming native domains of AI agents capable of operating without direct human involvement—although always under human supervision in sensitive contexts.

The case of SapFix, the tool developed by **Meta**, represents a milestone in **debugging** automation. This system generates patches for detected bugs, validates them through automated testing, and submits them for human review before deployment. What's relevant here is not only the automation of patching, but also the **smooth integration between AI and humans in a cycle of continuous improvement, traceability, and validation.** This type of hybrid collaboration lays the foundation for augmented reliability engineering, where humans supervise and fine-tune, but don't intervene directly in every incident.

Beyond debugging, this capability is reaching a new level of automation in systems management. These agents can **apply patches, restart services, scale resources, and run maintenance tasks** based on contextual conditions. In cybersecurity, they can even act as active sentinels, blocking threats in real time and ensuring regulatory compliance without requiring immediate intervention from the IT team.

In the support domain, AI is optimizing the entire incident response cycle. NLP systems can **classify tickets, prioritize them by criticality, and generate automated responses based on knowledge bases.** This frees up support teams and improves response times, while providing end users with a more agile and personalized experience.

Moreover, advanced platforms now enable these flows with conversational assistants and LLM agents capable of **interacting with users, understanding informal problem descriptions, diagnosing likely causes, and offering automated or guided solutions.** The result is a layer of distributed intelligence that transforms support into a conversational, comprehensive, and adaptive process.

Platforms like Retool with Agents or Amazon with Kiro already show that agents can autonomously complete more than 30% of complex tasks without any kind of supervision from the development team.

75%

identify AI-driven testing as key to their strategy for 2025

16%

of respondents have adopted AI in testing, showing that real adoption is still lagging

5 Reverse engineering and automated refactoring

Modernizing legacy software is one of the most urgent challenges. Thousands of lines of code remain in operation in outdated languages.

The main catalyst for this change is the **ability of LLMs to understand and generate code in multiple languages,** as well as infer structures, patterns, and functionalities from codebases without documentation. These capabilities enable new ways to approach modernization: from automated function analysis to semantically precise translation into modern languages, including vulnerability detection and the reconstruction of lost technical knowledge.

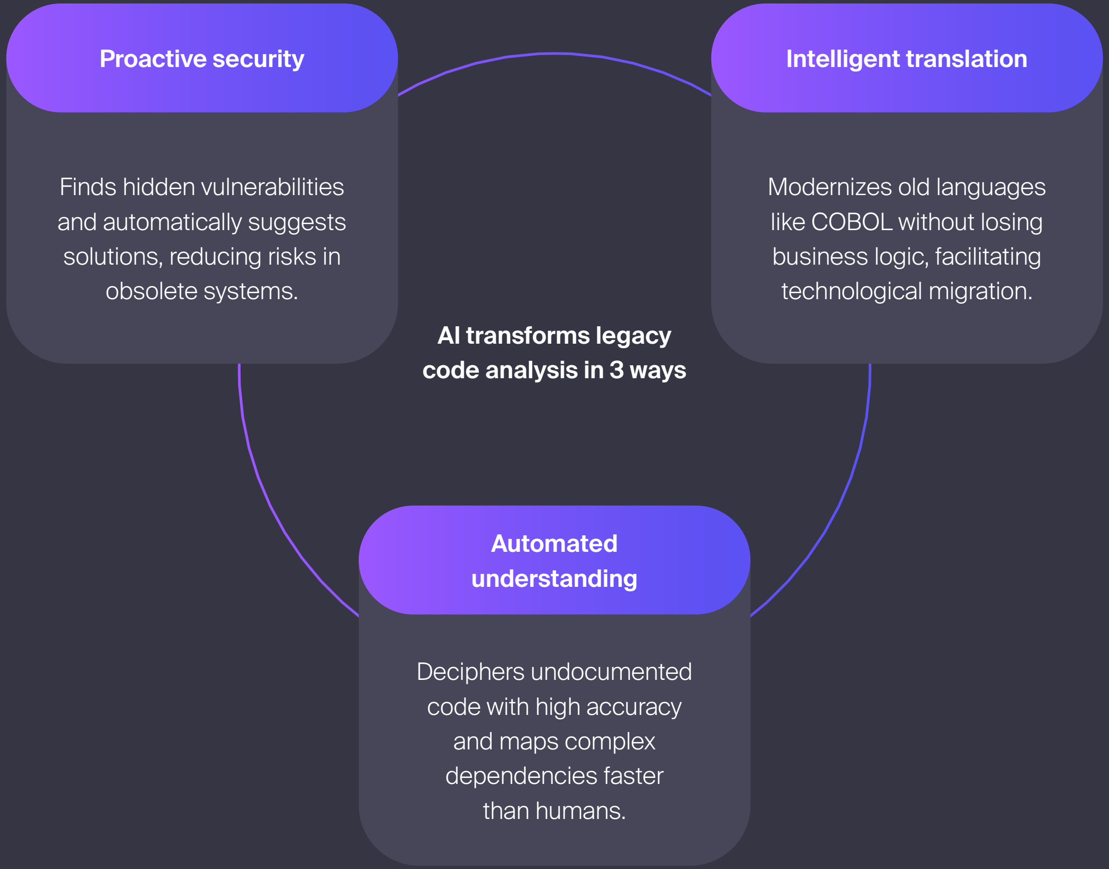
The organizational impact of these tools is equally significant. Instead of engaging in costly and risky “big bang” migrations, **companies can adopt progressive modernization strategies supported by artificial intelligence.** Examples include Japanese banks that have converted millions of COBOL lines to Java in under a year, or the U.S. government migrating unemployment systems from PL/I to Python—showing this promise is real and already underway.

At the technical level, **AI is transforming reverse engineering in at least three dimensions.**

First, through automated code understanding, with models able to identify function purposes, module dependencies, and complex control structures. Second, through the intelligent translation of code, where the focus is no longer just syntactic conversion, but maintaining the context and logic of the code. And third, by strengthening the security of these systems—detecting historical vulnerabilities and proposing remediation aligned with modern cybersecurity standards.

This automation is also reaching hostile code analysis. Models like DeGPT, designed to improve the readability and simplification of decompiled code, **enable rapid understanding of malicious software behavior, mapping it to frameworks such as MITRE ATT&CK** and generating mitigation recommendations. In this field, AI not only saves time; it represents a new defensive layer against advanced threats.

In the U.S., more than 80% of public IT spending is still allocated to maintaining old systems, which highlights the urgency of automating modernization. Tools developed by IBM have shown the ability to **maintain more than 90% functional coverage with minimal intervention.**



Proactive security

Finds hidden vulnerabilities and automatically suggests solutions, reducing risks in obsolete systems.

Intelligent translation

Modernizes old languages like COBOL without losing business logic, facilitating technological migration.

AI transforms legacy code analysis in 3 ways

Automated understanding

Deciphers undocumented code with high accuracy and maps complex dependencies faster than humans.

40%
of legacy system modernization projects will incorporate AI-powered reverse engineering by 2026.

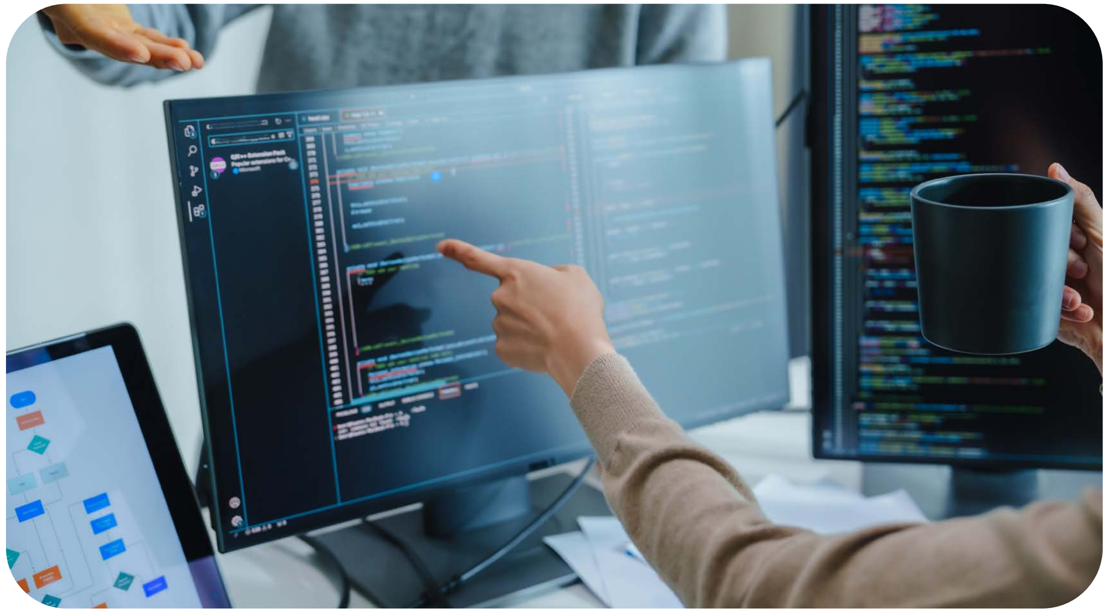
83%
is the accuracy with which AI identifies the purpose of undocumented functions in legacy C code, outperforming junior developers by 67%.

6 Vibe Coding & VibeOps: vibe-driven programming

At the beginning of 2025, a concept was introduced that quickly captured the attention of the tech community: **Vibe Coding. This trend redefines programming as an activity guided by the developer's intent**, who expresses in natural language what they need, while LLM models translate that intent into executable code.

Shortly after, its natural evolution emerged: **VibeOps. An AI-driven operational approach that seeks to maximize developer productivity by eliminating operational frictions and allowing them to focus on value creation.**

Both trends **share a common vision of transforming the development experience into a fluid, multimodal, and AI-assisted process,** where the developer becomes an architect of intentions rather than a performer of mechanical tasks.



Vibe Coding: rom intention to executable code

This approach prioritizes rapid creation over premature optimization or **"code first, refine later."** This trend aligns perfectly with agile frameworks, facilitating rapid prototyping, iterative cycles, and early concept validation. Thus, programming becomes a more dialogical and intuitive experience, where the developer maintains conceptual control and AI acts as a technical copilot.

The advantage lies in accelerating the path from idea to MVP, enabling low-cost technical exploration, greater adaptability, and faster pivoting, minimizing sunk costs. The goal is a development environment where AI agents act as omnipresent assistants—that is, they suggest in real time, automate tedious processes, and even generate the basic structure of complete applications. Tools already available point in that direction, such as environments like Replit or the Cursor IDE with integrated generative model support that enable this conversational flow with the machine for code creation.

A **vibe** developer can say out loud what they need, see the code generated instantly, run it, and if something fails simply describe the error to the AI for it to fix. Currently, this action is limited, but by 2030, with much more powerful and specialized LLMs, **it could become a standard development mode for prototypes and even for serious projects in certain circumstances.**

This requires new skills for developers, including clear communication, sharp judgment, and general-purpose thinking. LinkedIn's 2025 Jobs Report notes a **300% increase in demand for "AI Prompt Engineer" roles**, showing how vibe-driven coding is rewriting job descriptions.

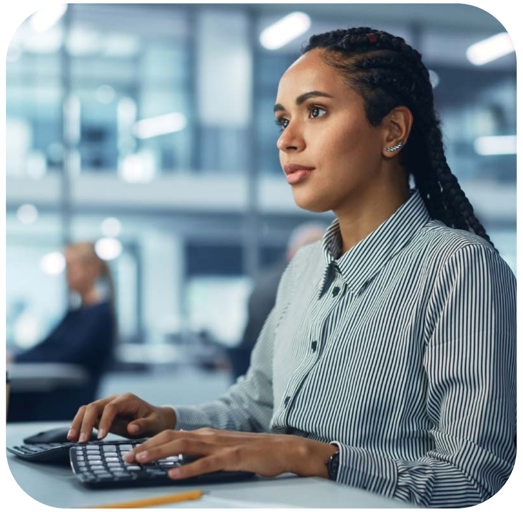
VibeOps integrates intelligent operations into the development flow, eliminating friction and enabling uninterrupted creativity

VibeOps: invisible operations, visible productivity

VibeOps emerges as a natural extension of the vibe coding concept, introduced by Andrej Karpathy, where **the programmer guides, refines, and supervises code generated by AI without leaving their creative environment**. The leap proposed by VibeOps is even more disruptive: eliminating operational frictions (infrastructure, deployment, monitoring, maintenance) by delegating them to intelligent systems that respond to natural language instructions, directly from the code editor.

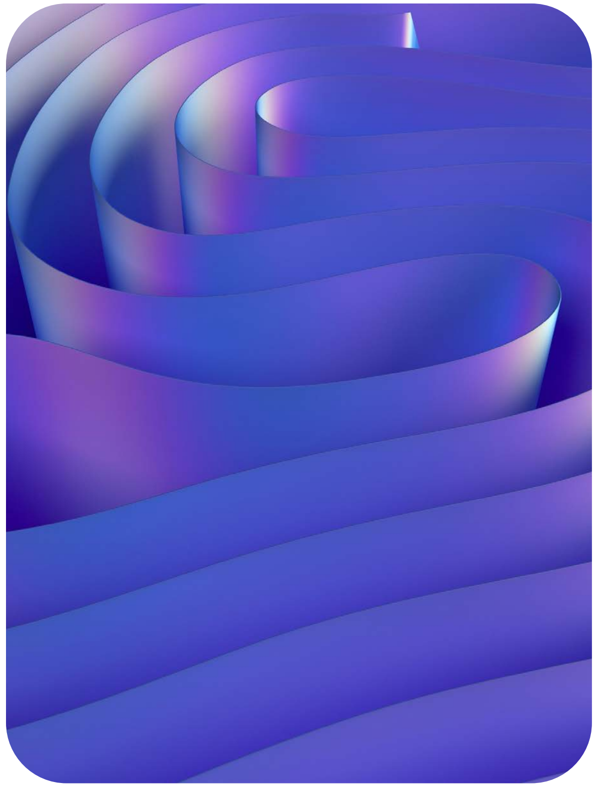
While DevOps requires teams to have specific knowledge in tools like YAML, Terraform, or monitoring scripts, VibeOps proposes a radically different experience.

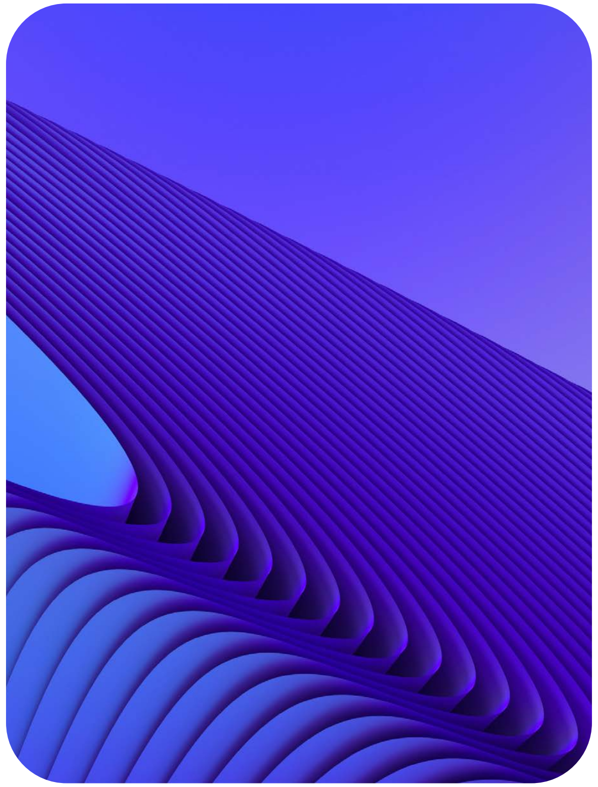
This approach transforms the developer's relationship with the operational lifecycle of software. It's no longer about collaborating with operations teams, but about **integrating operations within the development flow, without interruptions or the need for context switching.**

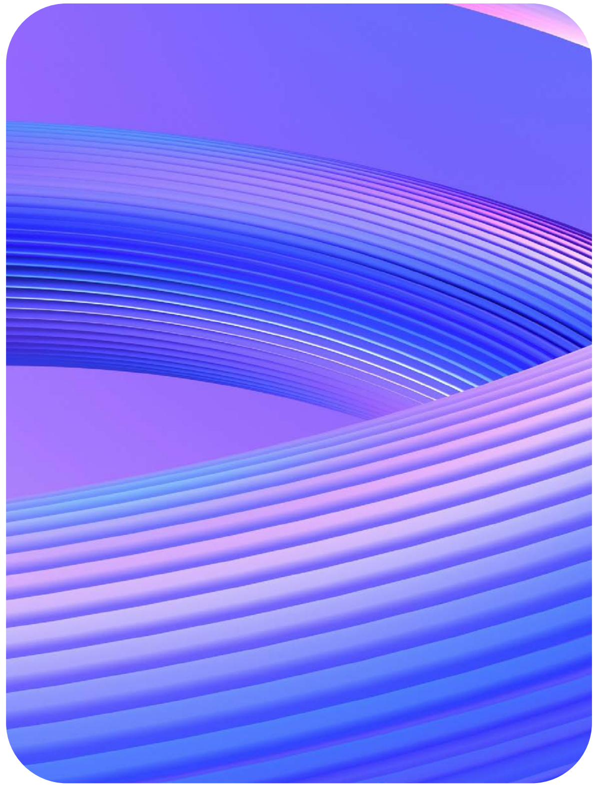


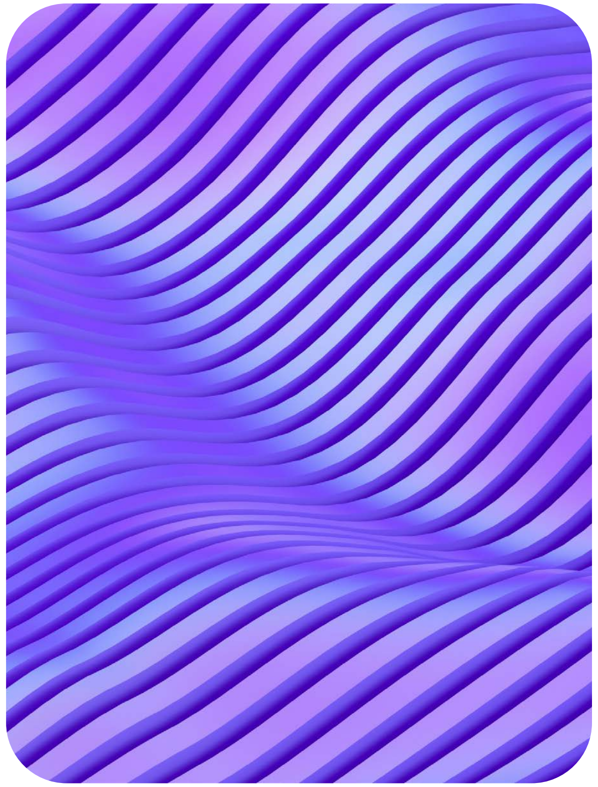
DevOps traditional	AI-powered VibeOps
Infrastructure as code	**Conversational infrastructure guided by AI**
Manual configuration	**Context-based autoconfiguration**
Complex CI/CD pipelines	**Instant deployment from the editor**
Monitoring scripts	**Automatic alerts, logs, rollback by default**
Specialized support	**LLM-assisted response in natural language**

This philosophy is based on four pillars:









Conversational infrastructure definition

Developers describe what they need in natural language; AI designs, configures, and implements directly from the development environment.

Assisted incident response

In case of errors, AI analyzes logs, diagnoses problems, and suggests solutions without the developer leaving their main task.

Development environment optimization

AI learns from user behavior to anticipate bottlenecks, adjust resources, and personalize the experience without direct intervention.

Platform continuity

Allows you to move code between local, staging, or production environments without a change in mindset or configuration, unifying the development experience.

VibeOps redefines success in engineering by focusing on metrics that enhance the developer's experience, efficiency, and creativity

Implementing VibeOps requires rethinking what is measured, how it's measured, and why. It's no longer just about capturing system performance data, but about creating visibility into the friction, blockers, and enablers that impact the developer's creative flow.

Unlike traditional approaches focused on system performance metrics (uptime, errors, throughput), **VibeOps introduces a new set of indicators centered on the developer experience.**

Reduction of context switching time (context switching)

Deployment frequency

Average time to create new environments (time to environment)

Duration of creative flow state

Cycle time from idea to production

These metrics are the compass guiding the implementation of VibeOps, aligning operations with the ultimate goal: **accelerating value delivery through a frictionless development experience.**

This trend does not replace the core principles of DevOps (collaboration, automation, continuous delivery) but rather expands them with a critical layer: the intentional design of the developer experience as a primary source of efficiency and creativity.

Impact of AI on software consulting: the future of engineering firms

Generative AI is revolutionizing software development, redefining the role of the developer as a code coordinator and evaluator

The transformation of consulting firms in the era of intelligent code

The impact being experienced by consulting firms driven by Gen AI and automation involves changes that are both strategic and operational. The delivery of one-off projects is being replaced by **models of strategic, active, and continuous partners, capable of anticipating and adapting to the pace of innovation.**

At the internal organizational level, the way software is conceived, developed, tested, and maintained is changing drastically, marked by **advanced automation**, a significant increase in productivity, and the consolidation of new collaborative frameworks.

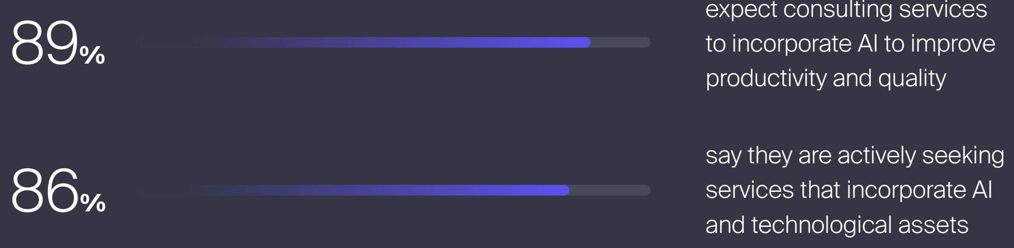
This shift has led to the **emergence of new professional trajectories.** The role of the developer is transitioning from code writer to reviewer and validator of results generated by AI. As LLMs gain precision and reliability, this profile evolves into that of a development orchestrator, responsible for supervising, adjusting, and ensuring the quality of automated outputs.

In parallel, **key debates are opening up about the future of work, the ethics of integrating AI, and the role of consulting** in a technological ecosystem dominated by intelligent platforms and autonomous solutions.

Moreover, the unpredictable nature of this technology entails the **emergence of new specialized roles.** It is no longer enough to master traditional programming languages; professionals are now expected to be able to select, manage, and integrate multiple AI models into existing systems.

In addition, **continuous learning and constant upskilling are becoming imperative.** Organizations must promote tailored training programs that include both technical skills and soft skills, such as effective communication and project management.

Consulting clients want AI-powered services



89% expect consulting services to incorporate AI to improve productivity and quality

86% say they are actively seeking services that incorporate AI and technological assets

Finally, beyond enhancing individual productivity, there is an opportunity to transform the collaborative development of software. When integrated into frameworks like Agile or DevOps, it optimizes workflows, anticipates failures, and strengthens coordination. Automation of testing, deployment, and planning relieves operational load, but it will **require reinforced traceability, validation, and ethical oversight to guarantee trust.**

Generative AI is driving software consultancies to reinvent processes, adopt new skills, and offer outcome-based services

Internal organization is not the only thing being transformed by artificial intelligence. **Current business models must undergo structural changes and shift towards higher value-added schemes**, where the focus is placed on the strategic and operational impact their services generate for the client.

Today, advanced tools such as automatic code generators (GitHub Copilot, Codeium, among others) are being integrated into workflows, allowing for faster delivery without compromising quality. But the shift goes further, as providers are beginning to position themselves as capacity providers, offering specialized services such as the personalization of AI models according to the client's context, quality audits of automatically generated code, or **consulting for the integration of artificial intelligence into existing digital products.**

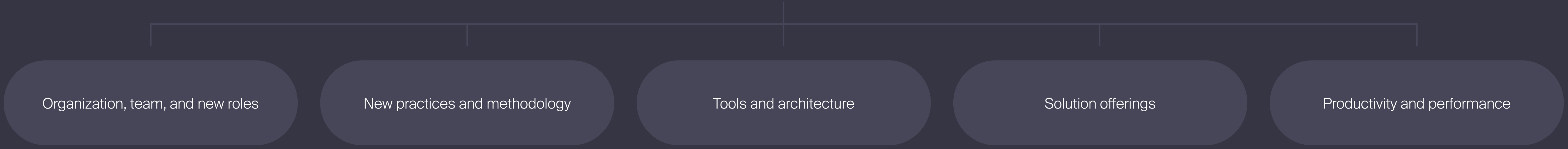
This transformation is accompanied by **greater emphasis on the creation of reusable intellectual property (IP) assets**, such as specialized frameworks, knowledge libraries, models trained for specific industries, or preconfigured solutions ready to adapt—thus increasing delivered value and reducing dependency on developments from scratch.

In addition, new monetization schemes more aligned with client outcomes are emerging. It is anticipated that, thanks to AI's ability to connect fragmented data and make the value generated end-to-end more visible, pricing models will tend to be outcome-based, where the client **will pay for the achieved KPIs or obtained benefits.**

This movement is also part of a transition from the traditional Software as a Service (SaaS) model towards an Artificial Intelligence as a Service (AIaaS) model, **where intelligent agents provide continuous support, learn from usage, and evolve without the need for manual updates**.

Altogether, these **changes require rethinking not only how software is developed, but also how value is captured and delivered.** Thus, by becoming strategic partners, it will be necessary to engage in four key areas: decision-making with AI, automation of internal tasks for greater scalability, responsible adoption of technologies through ethics and compliance, and the formation of multidisciplinary teams that integrate business, data, and technology capabilities.

Impact of the new era of AI-driven code and augmented engineering across different areas of the organization



Organization, team, and new roles

New practices and methodology

Tools and architecture

Solution offerings

Productivity and performance

Intelligent collaboration drives new strategic and technical roles, boosting productivity, quality, and innovation

Human–machine collaboration that redefines the role of the developer

The application and implementation of artificial intelligence is comprehensively reshaping the experience of software developers—from how software is written to how it is conceived, built, validated, and delivered. This transformation involves a rethinking of the workflow, team roles, and collaboration between different areas. As a result, **the developer becomes a capacity orchestrator,** combining AI resources with human knowledge to build solutions faster and with greater technical coherence.

AI is also playing an active role in the definition of user stories, acceptance criteria, and functional requirements, enabling development teams to convey this information more clearly to business analysts and stakeholders.

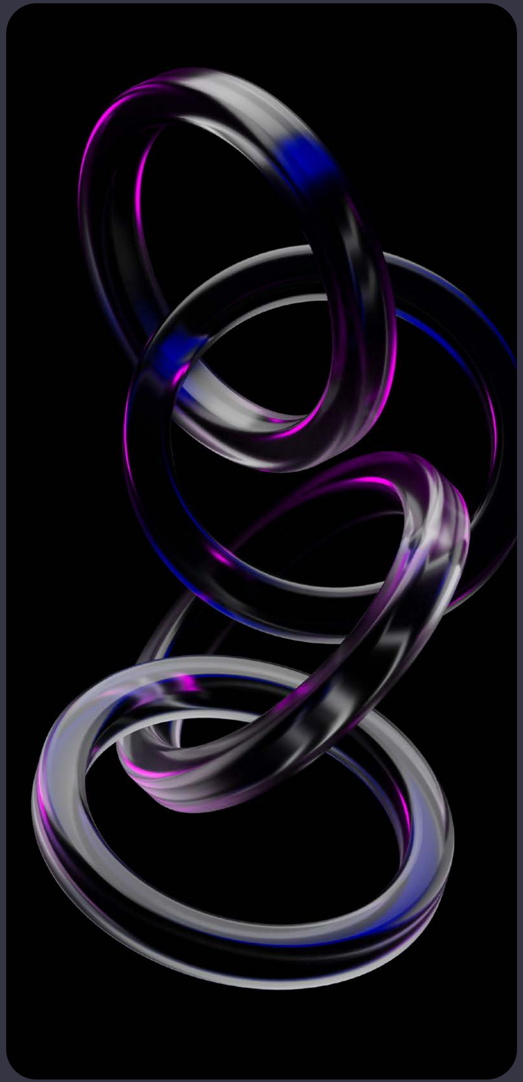
In terms of interface design, **AI can define base structures for pages, navigation flows, and functional components.** From these structures, designers step in to shape the interactive layers and more complex user experiences, ensuring intuitive navigation centered on the end user.

Moreover, it is paving the way for true continuous delivery. Agile teams can use it to generate large volumes of code, write pull requests (PRs), and maintain a steady flow of them. **This accelerates time-to-market and turns the development cycle into a much more fluid process,** reducing wait times and task dependencies

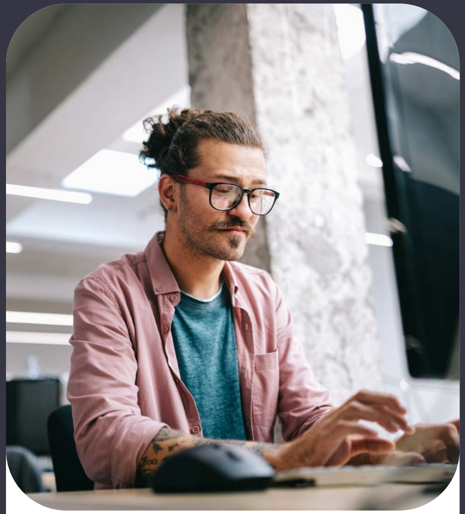
The emergence of collaboration as a turning point

This impact generated by AI, along with its democratization and professionalization, extends beyond pure software development. **The new collaborative model also opens the door to emerging professional career paths**, such as Prompt Engineer, AI Trainer, Data Steward, or Platform Engineer, among many others. Many of these profiles are being formalized by IT consultancies and other specialized companies leading the deployment of advanced solutions and needing to ensure the proper integration between human capabilities and artificial intelligence.

That is why a**daptability, continuous learning ability, and interdisciplinary competencies are increasingly valued.** Companies that invest in hybrid teams will tend to innovate faster, reduce errors, and attract next-generation talent interested in working with cutting-edge technologies.



The new era of code is redefining teams and consulting, shifting from execution to strategy, automation, and innovation



Prompt Engineer

Specialist in designing instructions for generative models, tailoring their responses to specific contexts. Accelerates prototype creation and ensures useful and controlled results.

AI Software Architect

Orchestrates the ecosystem required to bring AI into production, aligning data science, business, and regulation. Key to scaling projects with structural guarantees.

MLOps Engineer

Automates and manages the AI lifecycle, ensuring performance, traceability, and compliance. Acts as the bridge between data science, engineering, and operations.

AI Ethicist & Risk Analyst

Assess social impacts, biases, and regulatory risks of AI, defining ethical and governance frameworks for responsible and compliant development.

Data Steward & AI Trainer

The former ensures data quality and governance; the latter trains and fine-tunes models to ensure their responses are relevant, ethical, and adapted to real-world context.

From staff augmentation to capability augmentation and new pressures

The staff augmentation model, based on increasing the number of external developers or analysts to cover demand peaks or compensate for skill gaps, is being replaced by an approach focused on capability augmentation. Instead of adding people, **organizations seek to multiply their effective capacity through the strategic use of, for example, AI tools, process automation, intelligent copilots, specialized frameworks, and assisted development platforms.** In this way, development teams become smaller, but more specialized, capable of generating greater value.

This shift toward capability augmentation has direct effects on the talent profiles organizations demand. Tasks that traditionally fell to junior roles—such as writing basic functions, running unit tests, researching documentation, or analyzing data—are now being handled by intelligent systems, creating structural pressure on entry-level positions. From now on, **even junior profiles are expected to understand how to collaborate with AI and how to generate value from the very beginning.**

Simultaneously, **new hybrid profiles are gaining prominence**—professionals capable of interpreting AI-generated outputs, refining prompts for better results, making data-driven decisions, and providing critical insight into AI proposals will lead to more specialized teams, where the operational layer will be automated and the consultative and strategic layer will be strengthened.

With LCNC, citizen developers emerge, opening up software development to non-technical profiles

The new figure of the citizen developer

The rise of LCNC platforms has driven the emergence of the citizen developer. These are **professionals without technical backgrounds who, thanks to these tools, create functional applications to solve business needs with greater autonomy and speed.**

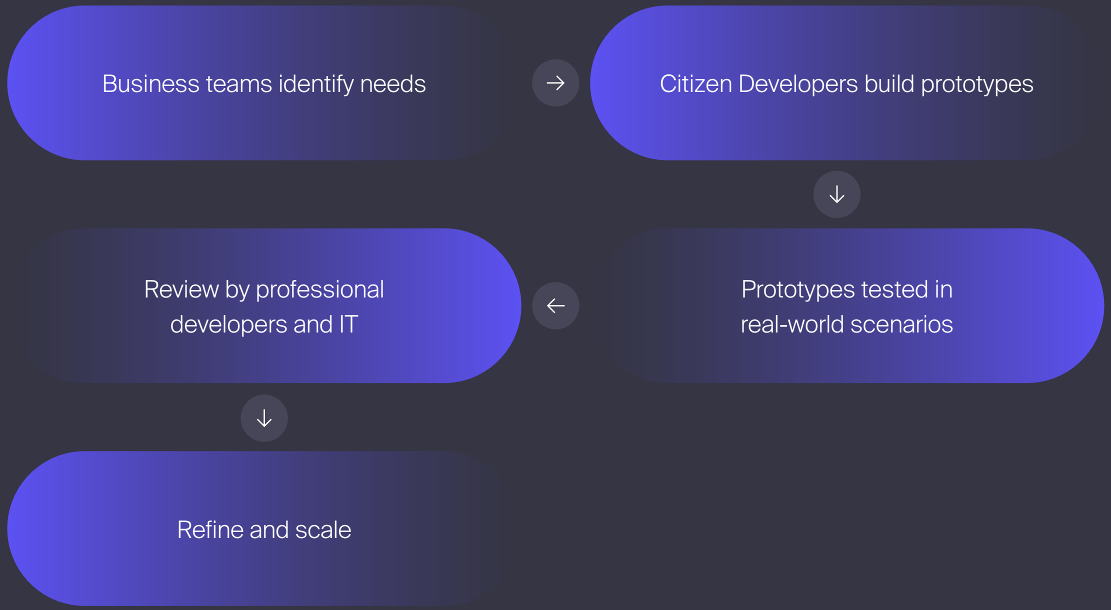
Companies are investing in these new professionals in search of benefits that are already tangible. Citizen developers have reduced application delivery times by up to 70% and development costs by 50%.

Moreover, **it is estimated that 30% of automation applications with generative AI will be created by citizen developers in 2025.** Therefore, it is expected that, along with professional developers, they will form an agile ecosystem where business knowledge is combined with technical robustness. All of this, framed within good governance, will guide citizen developers and provide support for innovation while minimizing the potential risks of implementation.

They are distinguished by a set of **characteristics** that enable them to create applications without being programming experts:

- **They do not require advanced technical knowledge:** they rely on low or No-Code platforms with intuitive graphical interfaces to build applications, instead of writing complex code.
- **Focus on solving business problems:** they are professionals from other functional areas who identify concrete needs in their daily processes.
- **Deep business knowledge:** unlike traditional developers, citizen developers understand firsthand the processes and objectives of their department, allowing them to perfectly align the tech solution with real operational needs.
- **Multidisciplinary collaboration:** citizen developers tend to work collaboratively, acting as a bridge between the business area and the technical area.

Citizen developer workflow model


Business teams identify needs
Citizen Developers build prototypes
Prototypes tested in real-world scenarios
Review by professional developers and IT
Refine and scale

98%
of companies use platforms, tools, or Low-Code features in the development process.

84%
say that Low-Code enables more people to participate in the application development process.

75%
believe that Low-Code is the only option for programming in the future.

A new approach emerges: fewer lines, greater impact — driving fast, collaborative development with tangible results

Strategic relevance of citizen development

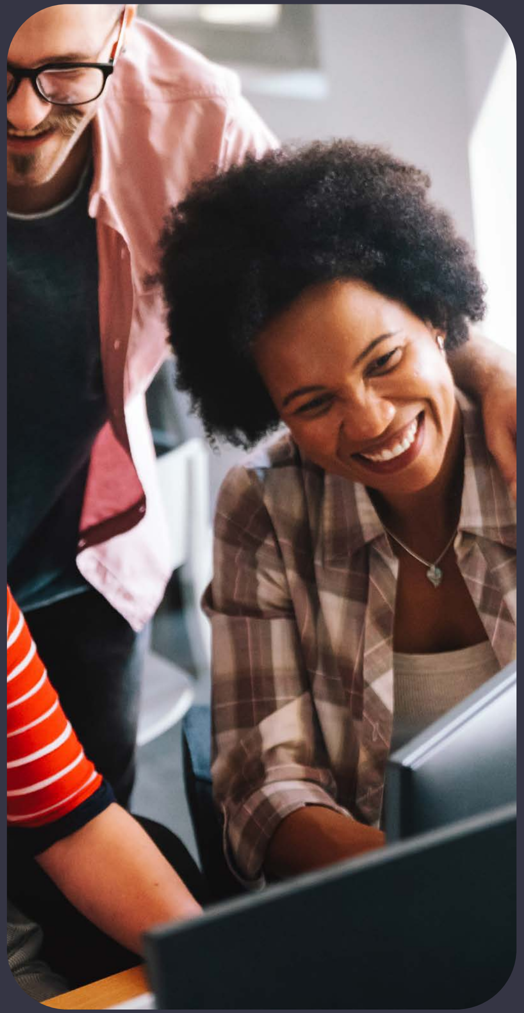
Acceleration of digital transformation	By enabling more people to contribute to solution creation, citizen development democratizes innovation within the company.
Reduces costs and time	LCNC platforms minimize the need for complex developments, which lowers IT costs and speeds up application delivery times.
Improves productivity	These employees tend to be more engaged and motivated, as they participate directly in the creation of tools that facilitate their work.
Relieves the it department's burden	By delegating to business areas, the creation of simple applications or automation of specific tasks is enabled.

A new vision with fewer lines and more impact

From this movement, **the philosophy of fewer lines, more impact is gaining momentum.** Seeking to achieve greater business results by writing the least amount of code possible—something LCNC platforms already make possible—modern tools allow a user to create through predefined components and visual flows, drastically reducing the amount of hand-written code.

In practical terms, it means that a person can build a useful application with just a few clicks and configurations, in significantly less time than with a traditional language. This efficiency can result in solutions deployed in days or weeks instead of months and tailored exactly to the identified need. Moreover, this paradigm not only accelerates development, but **can also reduce errors, since less written code often means fewer bugs, and facilitates maintenance.** Updates are made by adjusting parameters or replacing visual components, instead of having to refactor large sections of source code.

Thanks to Low-Code/No-Code platforms, organizations can achieve **great results in digitization and automation** with very little programming effort, empowering all employees to contribute to innovation.



New practices with AI and automation optimize development and delivery, boosting efficiency, quality, and rapid response to change

AI and automation redefine software development and delivery practices

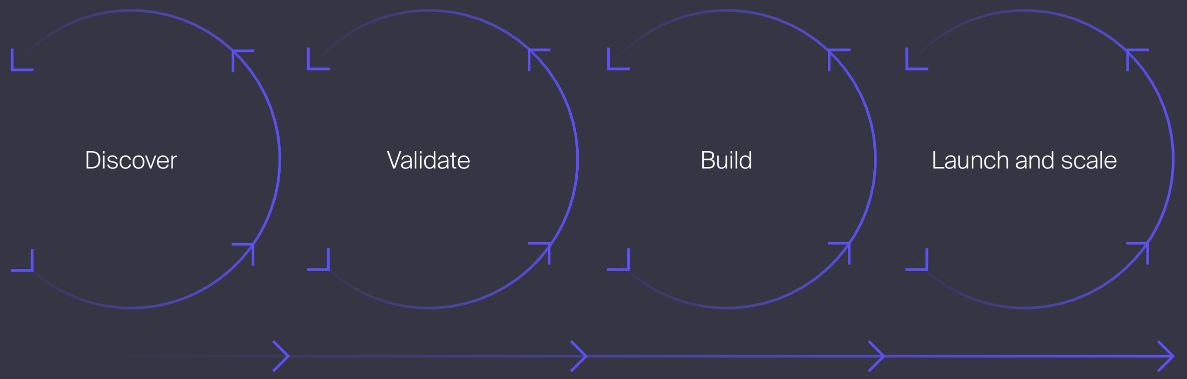
Beyond their role as point tools, artificial intelligence is **becoming consolidated as an operational layer that spans every link in the software development chain.** From the initial writing of code to its deployment in production and continuous evolution, it acts as an engine that optimizes flows, anticipates failures, and adapts solutions to the changing context.

Its value lies not only in accelerating individual tasks, but in enabling a more cohesive and intelligent ecosystem, where testing is automatically adjusted to user behavior, project management becomes predictive, and customer experience is personalized in real time. In this sense, **it becomes an infrastructure that makes possible a new way of building, deploying, and maintaining software** in increasingly complex and dynamic environments.

In fact, its integration is decisively accelerating software delivery. By automating a large number of tasks, its use reduces delivery times, improves software quality, and allows errors to be detected before they become critical issues, **turning the entire process into a proactive, reviewable, and efficient agile development,** which is key to sustaining continuous and ever-changing development cycles.

Reimagining the software product development lifecycle — from fragmented to accelerated with AI

Current software product development lifecycle



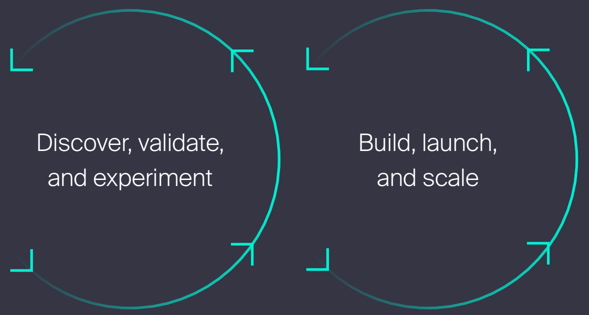
Dispersed and isolated data across research, usage, and success

Fragmented ownership across product, marketing, engineering, and outcomes

Long timelines and delays between phases

AI-enabled software product development lifecycle

Integrated customer data that enables insights across all horizons



Integrated real-time data that generates insights

Aligned ownership, product managers as “mini CEOs”

Radically compressed cycles, faster time to market

Integrating security from the start with Shift Left Testing reduces risks, lowers costs, and strengthens trust in the software

Shift left testing in security: integrating controls from the start

As a result of the evolution of software engineering driven by AI and automation, testing and security processes have undergone a significant transformation. In particular, organizations are adopting the "shift-left" approach in their DevSecOps practices, **moving security tests and validations to the earliest stages of the development lifecycle.** Instead of leaving these tasks for the end, they are now integrated from the design and initial coding stages.

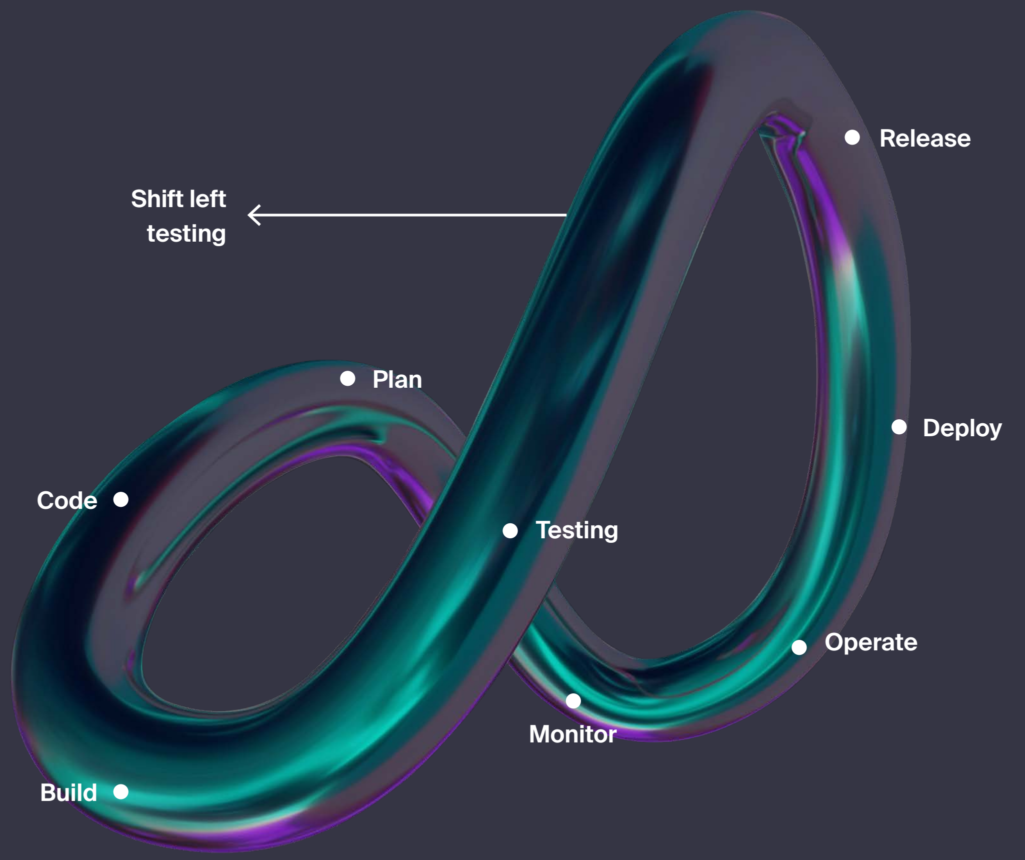
This strategic shift results in safer products, as it **enables the detection and correction of vulnerabilities from the start,** significantly reducing risks and strengthening software integrity. In addition, it involves considerable cost savings, since addressing security issues in later stages often requires costly rewrites and exposes organizations to sanctions from security breaches. At the same time, it raises the technical level of the team by involving developers in early stages.

Another impact is the promotion of cross-functional collaboration. "Shift left" encourages direct cooperation between developers and security specialists, facilitating more efficient communication, a smoother exchange of knowledge, and a shared understanding of the process.

Large companies, such as Google or Microsoft, are already using AI-powered tools to automate product testing, automatically generating test cases, prioritizing executions, and anticipating defects. Tools have also been developed that **use Machine Learning algorithms to perform fuzzing tests,** identifying vulnerabilities with greater precision and efficiency than traditional methods.

Finally, some streaming platforms already use **AI systems to monitor their streaming infrastructure in real time**, detecting anomalies, anticipating interruptions, and thus ensuring a stable and reliable user experience.

Shift left testing for high quality with delivery speed



Tools like Copilot and GPT are leading the assisted coding revolution, transforming both software productivity and quality

Enabling tools and technologies in software engineering

The generative AI market for software development, valued at $21.4 million in 2023, is expected to reach $117.6 million by 2033, with an annual growth rate of 19.1%. The emergence of assisted code generation tools powered by AI—such as GitHub Copilot, GPT, Tabnine, CodeWhisperer, or Replit—**has been one of the most disruptive innovations in this sector, enabling the translation of natural language instructions into functional code.** Most of these solutions, which combine natural language processing with deep learning, not only generate on- demand code, but also **actively scan the development environment, detect errors, and suggest improvements, learning from the team's context** as they are used. In doing so, development cycles are shortened, code quality improves, delivery speeds up, and the team's creative capacity expands.

Tools

ChatGPT

GitHub Copilot

Amazon CodeWhisperer

bolt

replit

tabnine

Faster development speed:
allowing teams to focus on new features.

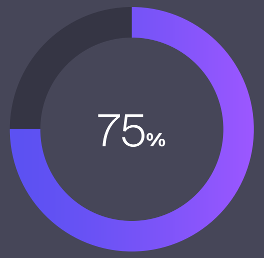
Automation of repetitive tasks:
freeing up time to tackle complex challenges.

Cleaner and more readable code:
with fewer errors and greater maintainability.

Accelerated learning for junior developers, thanks to real-time explanations and suggestions.

These tools have been shown to **accelerate task completion by up to 56%**. Developers who use them report higher levels of satisfaction: **between 60% and 75% say they feel less frustrated, more focused, and more satisfied with their daily work.** The impact is especially notable in mature organizations and among senior developers, who better understand how to leverage AI as an extension of their capabilities.

The benefit is not limited to immediate performance: **73% of developers who use GitHub Copilot say it helps them stay in a state of flow, while 87% say it helps them conserve mental effort** during repetitive tasks.



of developers already use AI tools in their workflow

Each phase of the software cycle benefits from augmented assistance, reducing errors and accelerating delivery with precision

Tools applied throughout the entire software development lifecycle

The transformation of software engineering in the era of artificial intelligence is the result of an ecosystem of enabling technologies, intelligent tools, and flexible architectures that act as catalysts for a new way of conceiving, building, and maintaining digital solutions. This technological environment is driving the rise of AI-assisted development and the expansion of LCNC platforms, **making possible a faster, more inclusive, and personalization-centered software engineering.**

Use of tools in each stage of the software development lifecycle

Planning	Design	Coding	Testing	Code Review	Pre-Release	Post-Deployment
• Sketches interfaces • Detects requirements from the beginning • Improves coordination among teams	• Generates layouts • Enhances UX • Automates usability testing • Reduces manual flows	• Provides real-time code suggestions • Enables automatic refactoring • Automates tasks	• Proposes tests • Detects bugs • Cleans up code in context • Optimizes validations	• Audits code in real time • Detects errors and vulnerabilities • Generates automated technical documentation	• Documents code and changes • Keeps it adapted and up to date • Facilitates handover and version control	• Automates generation of visual content • Personalizes digital experiences • Maintains consistency
AI UI Design Tool	AI UI Design Tool	AI-Assisted Code AI Pair Programmer	AI Pair Programmer	AI-Reviewed Code AI-Powered Documentation Platform	AI-Powered Documentation Platform AI for Creative Tasks	AI for Creative Tasks

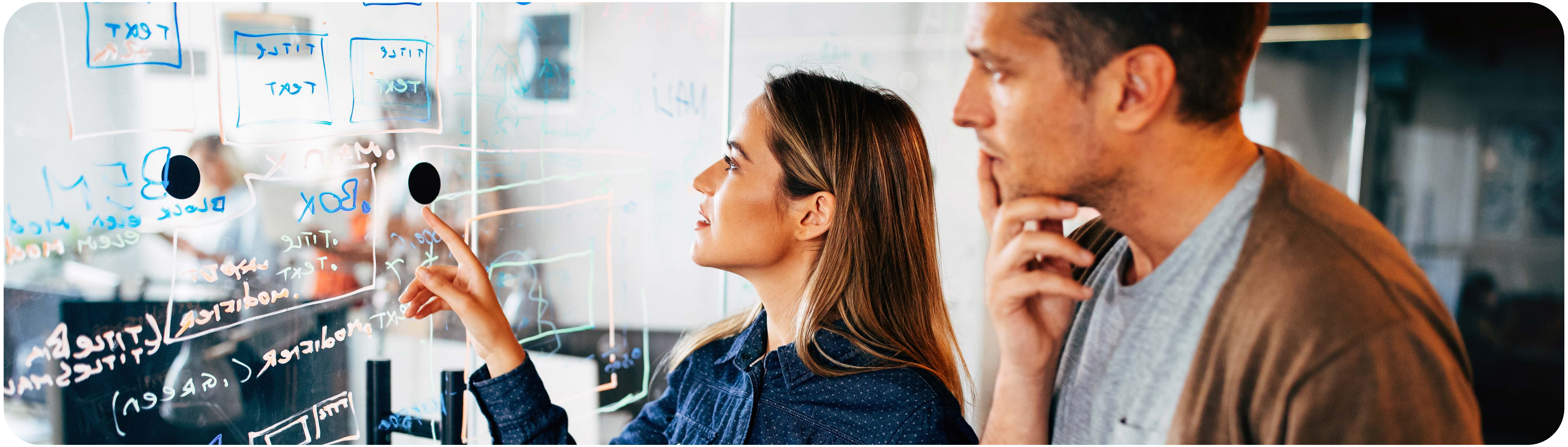
Tools

A modular and intelligent architecture turns every tool into a smart ally and every workflow into an opportunity for optimization

Support architecture in augmented software engineering: an intelligent and modular infrastructure

The support architecture behind the enabling technologies and AI tools in the software development lifecycle and augmented engineering is a **complex and modular framework that integrates platforms, services, protocols, and automation layers.**

This architecture enables AI solutions to be scalable, secure, collaborative, and adaptable to different development stages and needs. Moreover, its design aims to facilitate joint work between humans and machines, from ideation to continuous maintenance.



Characteristics of the support architecture

Modularity and reusability:

this architecture is based on decoupled components that can adapt to the needs of each team, product, or client. This allows agility to be maintained even in complex environments and facilitates constant evolution.

Embedded intelligence:

AI does not operate as an external layer, but acts as an integral part of the flow with copilots assisting in the IDE, recommendation engines, systems that automatically adjust pipelines, and models that learn from team behavior.

Scalability and flexibility:

designed to grow with the project, it adapts to large or small teams, simple or multi-level projects, cloud, on-prem, or hybrid environments.

Seamless integration:

various platforms allow connection, automation, and optimization without interruptions. Interoperability between systems is a technical priority that defines the developer experience and operational robustness.

Typical components of the AIASE stack

AI/ML Modules	code generation, refactoring, error detection, failure prediction.
Flow Orchestrators	connect development, testing, deployment, and monitoring.
Smart IDE	code assistants, contextual prompts, productivity metrics.
Observability Platforms	continuous monitoring of DevEx and software performance.
IaC Controllers	infrastructure defined by code using natural language.

Its strategic and operational impact

A well-designed support architecture is not visible to the end user, but it radically determines the success of development teams. **Its impact is reflected in shorter development cycles and faster deliveries, the reduction of manual errors and increased reliability, higher developer satisfaction (DevEx)** and in the sector's growing innovation.

The IT offering is being transformed by AI, positioning consulting firms as key agents of innovation, efficiency, and continuous support

Evolution of it consulting solutions in the new era of augmented software engineering

As these technologies become more accessible and complex, it becomes clear that a real challenge lies in **choosing the right stack, governing data, adapting architecture, developing internal talent, and aligning it all with business goals.** At this point of convergence between vision and execution, between architecture and culture, the role of consulting firms becomes crucial.

From the initial design and implementation of modern architectures (Cloud, hybrid, microservices, DevSecOps, among others) to the integration of AI, automation, or LCNC platforms, the involvement of IT consulting firms **enables maximum ROI** from these technologies, accelerating delivery cycles and mitigating operational and regulatory risks.

Particularly in environments where AIASE tools are implemented, **expert guidance helps solve structural challenges** that span the lifecycle of models (versioning, retraining, updates), the selection of open or closed source technologies depending on the use case and regulatory context. Moreover, these firms play a key role in ensuring traceability, interoperability between legacy and modern systems, and in establishing secure testing environments before scaling solutions to production.

From now on, **their value proposition no longer revolves around custom development or one-off outsourcing, but is structured as a comprehensive, modular, and ongoing offering**, capable of responding to the multiple technical, cultural, and strategic challenges faced by organizations competing in the digital economy.

Strategic intervention areas in digital transformation processes

Technological Diagnosis and Strategic Planning

They audit existing systems, assess digital maturity, and define roadmaps aligned with business objectives.

Solution Architecture and Technology Modernization

They design modular and secure architectures, modernize legacy systems, and implement platforms that enhance developer experience and technical governance.

AI Adoption and Intelligent Automation

They deploy customized AI solutions, implement MLOps pipelines, and boost the software lifecycle with AIASE tools, testing, and task automation.

Training, Change Management, and Cultural Evolution

They offer technical and strategic training programs, support cultural transformation with sustainable models, and foster internal talent development.

Project Management and Ongoing Support

They orchestrate digital transformation with agile PMOs, specialized support, and offer services under the CaaS model.

Applied Innovation and New Business Model Design

They explore emerging technologies, develop user-centered digital products, and create internal capabilities through innovation hubs and agile validation and scaling dynamics.

When AI unleashes talent, effectiveness stops being a goal and becomes the standard

AI-enhanced productivity that drives remarkable results

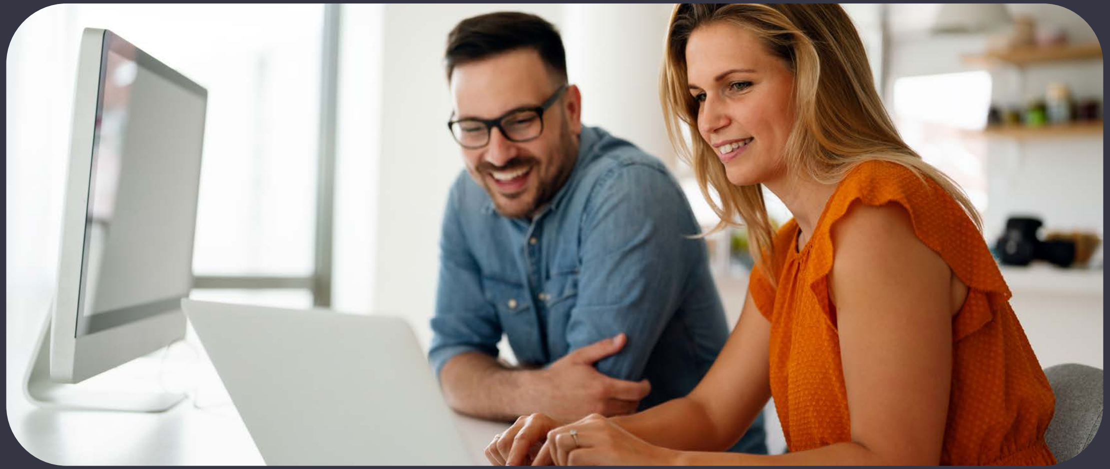
Tasks such as writing new code, refactoring, or documenting can be completed up to twice as fast. Specifically, code documentation and new code creation are shortened by around 50%, and code refactoring is performed in two-thirds of the usual time.

It's important to note that this impact decreases for complex tasks or when performed by junior profiles. In such cases, time savings may be limited. For example, some users took 7% to 10% longer with AI tools due to a lack of mastery in technical environments.

Conversely, experienced developers using AI-supported tools in complex tasks are up to 30% more likely to complete their work on time, thanks to the support these tools provide in exploring new languages, frameworks, or unfamiliar structures.

20-45%

the direct impact of AI on software engineering productivity, expressed as a percentage of current annual spending in this function.



Effectiveness in delivery

The effectiveness of software solution delivery **depends on multiple critical factors: code accuracy, product and functional quality, deadline compliance, and continuous adaptation capability.**

Previously, teams dedicated considerable effort to repetitive tasks such as base code writing, manual error detection, operational management, and deployment coordination. This not only slowed down development cycles, but also exposed projects to human errors, quality issues, and delivery delays.

Today, AI integration enables teams to automate many of these tasks, completely reshaping this landscape. Thanks to automation, the reinforcement of quality standards, and optimization of end-to-end processes, developers and engineers can operate with greater agility and reliability. **This frees up talent to focus on higher-value tasks** (such as innovation, design, and decision-making), achieving greater impact with less operational effort

The key to effective deliveries is combining AI with rigorous testing, security, and team adaptability

Among the most notable advances in terms of effectiveness are:

Real-time automatic error detection

Automated test generation that mitigates failures before reaching production

Optimization of code performance to maximize efficiency

Strengthening of security through proactive identification of vulnerabilities

Even with progress, **the full integration of AI into software development faces significant challenges**, as there are still obstacles such as the accumulation of technical debt from code generated without maintenance, contextual errors from AI, or its limited capacity to understand the particularities of each organizational environment. Added to this are concerns about privacy, data security, and regulatory compliance, which remain major barriers.

Finally, the compatibility of AI tools with existing technological ecosystems is not always immediate, which complicates smooth adoption. On the other hand, trust in the quality of code generated by AI has **increased substantially,** but risks related to reliability still persist—risks that can be mitigated with test protocols and validation mechanisms to ensure that the generated code meets both technical and business standards. Only then will it be possible to scale its benefits without compromising the integrity of the final product.

In conclusion, **organizations that choose to adopt augmented engineering face challenges** that go far beyond technological deployment. They will require expert support to select the right stack, adapt the architecture to regulated or complex contexts, govern the life cycle of AI models, and ensure seamless integration with legacy systems. This is why consultancies are shifting toward a capacity-based approach, offering continuous, strategic, and results-oriented support.

Productivity has been significantly enhanced thanks to code copilots, automated documentation generation, AI-assisted refactoring tools, and intelligent testing. However, **this positive impact depends on expert and well-contextualized use**, as in immature environments or inexperienced hands, AI can slow down the process or generate technical debt. Effectiveness in delivery has also been redefined. By relieving teams of workload, improved product quality and increased motivation and talent retention have been observed.

That said, **confidence in the benefits of AIASE must be accompanied by validation structures, quality protocols, and strategic vision.** Only organizations that approach this transition as a process of organizational maturity will be able to scale their capabilities without compromising technical integrity or the value of the final product.

The widespread adoption of AI involves critical risks such as technological dependency, algorithmic bias, and a high environmental impact

KPIs and engineering velocity in the era of AI-assisted development

This transformation of the role of IT consultancies, productivity, and the effectiveness of results **has a direct consequence on how the performance of development teams is measured.** Traditionally, teams have relied on indicators such as delivery speed, cycle time, or test coverage. However, in AI-assisted environments, these KPIs are evolving to reflect new dimensions in quality, efficiency, and impact.

Firstly, **specific metrics for AI usage are emerging,** such as model accuracy, inference latency, or throughput in production environments, which are key when deploying code assistants, test generators, or automated documentation. At the same time, intelligent code analysis tools allow teams to measure complexity, identify problematic patterns, and automatically assess maintainability, raising the quality bar from early stages.

Indicators are also expanding toward the developer experience, such as reducing cognitive load, improving focus, or supporting a sustained flow state, which are critical aspects to maximize the team's sustainable performance.

For their part, the concept of **engineering velocity will be redefined beyond the number of tasks completed**, with greater importance placed on the real value delivered and how AI enhances the team's autonomy, collaboration, and decision-making.

Automation accelerates execution, but the real benefit lies in how humans and intelligent agents coordinate.

Now, both speed and the balance between speed, quality, team well-being, and added value must be measured. **Adapting metric systems to measure AI's contribution is essential for those seeking to remain competitive** without compromising technical and human sustainability in software development. Here, consultancies play a key role in helping organizations incorporate AI tools and redesign their architectures, workflows, and—most importantly—their measurement systems.

Because without a measurement system that correctly interprets this impact, organizations risk being diluted between complexity and operational inertia. **Measuring well is, today more than ever, a competitive advantage.** Doing so intelligently—not just artificial intelligence, but also organizational intelligence—is what will make the difference between adopting AI and truly transforming with it.

How consulting firms can lead the new era of code?

Consulting firms are reinventing themselves, combining Low-Code, technical expertise, and continuous collaboration to lead sustainable digital transformation

How are consulting firms adapting to this trend without becoming obsolete?

Far from being displaced, consulting firms are evolving to capitalize on the potential of AI and position themselves as trainers and strategic partners in the implementation of these platforms. If their value proposition previously focused on providing technical teams and professionals to develop custom software, now they **must take on a more mature role, guiding, integrating, and scaling their clients' digital development.**

The new paradigm opens up new opportunities for those who know how to adapt with strategic vision. The challenge is no longer to compete in writing code, but to **lead the orchestration of agile, secure, and scalable development within each organization.**

Consulting firms that understand this new logic will be able to evolve within this growing sector paradigm. Those that persist in competing via "body shopping" and hourly billing, on the other hand, risk being overtaken by a more agile, decentralized, and empowered ecosystem.

This transformation involves five major lines of adaptation:

Governance, training, and scalability enablers

Act as strategic allies and trainers, creating Centers of Excellence, standardizing methodologies, and establishing best practices to avoid shadow IT.

Experts in advanced integration and the technical last mile

Position themselves as responsible for ensuring quality, security, and performance, complementing the capabilities of citizen developers.

Partners in continuous evolution

Shift from one-time software delivery to continuous improvement cycles, establishing long-term collaborative relationships that allow them to support clients through their evolution.

Experts and creators of reusable intellectual property

Convert their sector expertise into reusable modules or templates, positioning themselves as creators of intellectual property, not just providers of on-demand code.

Agents of change and organizational mentors

Lead cultural transformation by redefining flows, collaboration, and innovation, reducing risks, and accelerating conscious adoption of these technologies.

A new collaboration model, with consulting firms as drivers of intelligent and sustainable development

Example of the journey of continuous support and shared vision

1 Kickoff

The consulting firm does not expect detailed requirements but **rather co- creates the vision with the client.**

- Identifies opportunities to integrate AI
- Aligns business goals with tech strategy
- Defines innovation, productivity, and efficiency indicators

2 Design and Architecture

A **modular, scalable, and intelligent architecture** is designed.

- Allows for native integration of AI, automation, and observability
- Supports rapid iteration and continuous adaptation
- Applies "AI by design" principles such as self-healing and pattern recognition

3 Development and Testing

Code assistants, automated testing, and intelligent validation are incorporated.

- Speeds up development cycles and reduces repetitive tasks
- Improves the quality of AI-generated code through review and evaluation
- Aligns technical logic with business needs through effective prompts

4 Deploy/ Observability

Implementations are **automated and monitored in real time.**

- Configures autoscaling, proactive monitoring, and intelligent alerting systems
- Ensures full visibility of developer performance and experience
- Enables low-risk production experimentation

5 Evolution/ Feedback

The consulting firm remains after launch to continuously optimize and evolve.

- Adjusts tools and workflows based on real usage
- Optimizes the developer experience and reduces interruptions
- Integrates new technologies or needed functionalities

6 Culture and Innovation

The consulting firm drives a culture of augmented engineering and continuous innovation.

- Promotes AI-, automation- and agility-centered practices
- Trains teams on new tools and workflows
- Helps build sustainable internal capabilities

7 Business Growth

The consulting firm acts as a strategic guide for growth and technological efficiency.

- Advises on scalability, energy efficiency, and automation
- Introduces concepts like GreenOps, VibeOps, and intelligent DevEx
- Helps the client position themselves as an industry innovation leader

This massive adoption entails a number of risks that must be taken into consideration and proactively managed

This widespread adoption entails a certain number of risks that must be taken into consideration and managed proactively

Technological dependency and skill erosion

As developers rely more heavily on AI tools, there is a danger of over- dependence. This can **create operational vulnerabilities in the event of failures or disruptions** in provider platforms. Additionally, it may erode skills within technical teams, especially in more junior roles, turning teams into “less experienced professionals.

Algorithmic bias and hallucinations

These systems learn from historical data that may contain biases— whether in code quality, style, or even ethical or discriminatory content—posing the **risk of reproducing errors or producing unsafe results**. Furthermore, AI “hallucinations” represent a significant threat in software development. Indeed, it has been documented that many suggestions are occasionally irrelevant or incorrect.

Sustainability and carbon footprint

The most efficient code models consume approximately 50 tons of CO_2, despite being trained on energy-optimized infrastructures. Therefore, under increasing regulatory and social pressure, organizations must adopt more sustainable approaches, such as reusing already trained models, using renewable energy, or designing more efficient algorithms.

Leading transformation requires partners with vision, technical expertise, and strategic insight—not vendors focused solely on execution

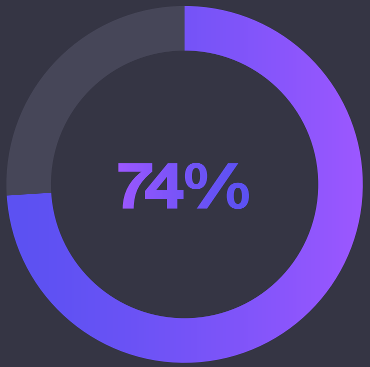
Take action now: lead the new era of code

The transformation shaping software development is not limited to new tools; it **implies a major structural shift redefining the foundations of technology delivery**—from team configurations to governance mechanisms and business relationship models. That's why it is not enough to merely adapt. Organizations that aspire to lead must anticipate change more rapidly and profoundly reconfigure how they create. Competitive advantage will not come from access to technology but from the ability to turn it into a driver of sustainable, differentiating change.

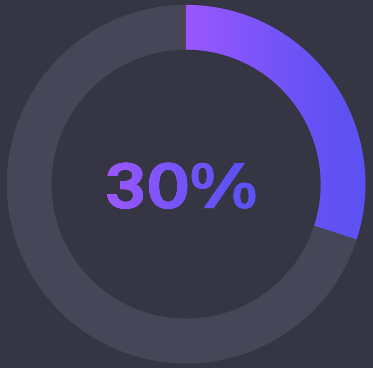
This is why **having a strategic and technological partner is a condition and prerequisite just to compete.** The new collaboration model is not transactional, but transformational. The role of the partner goes beyond project execution to become an enabler of the future—combining technological advisory, fluid integration of new tools, and strategic vision, while translating complexity into opportunity, integrating capabilities the client cannot scale internally, and supporting transformation with a comprehensive, agile, and sustainable approach.

Their contributions are varied: from assistance with advanced capabilities, specialized talent, smart methodologies, and continuous access to global or future trends, to guidance on regulatory and ethical frameworks. This partnership **enables process optimization, accelerated innovation cycles, reduced operational costs, strengthened digital resilience, and the scaling of initiatives without overburdening internal capabilities.**

Having a strategic partner means having a constant guide in an environment where change is fast and competitive pressure leaves no room for error. Consultancies not only help implement technologies like AI—they also act as catalysts for innovation, anticipating market trends and **helping companies build business models ready for the future.**



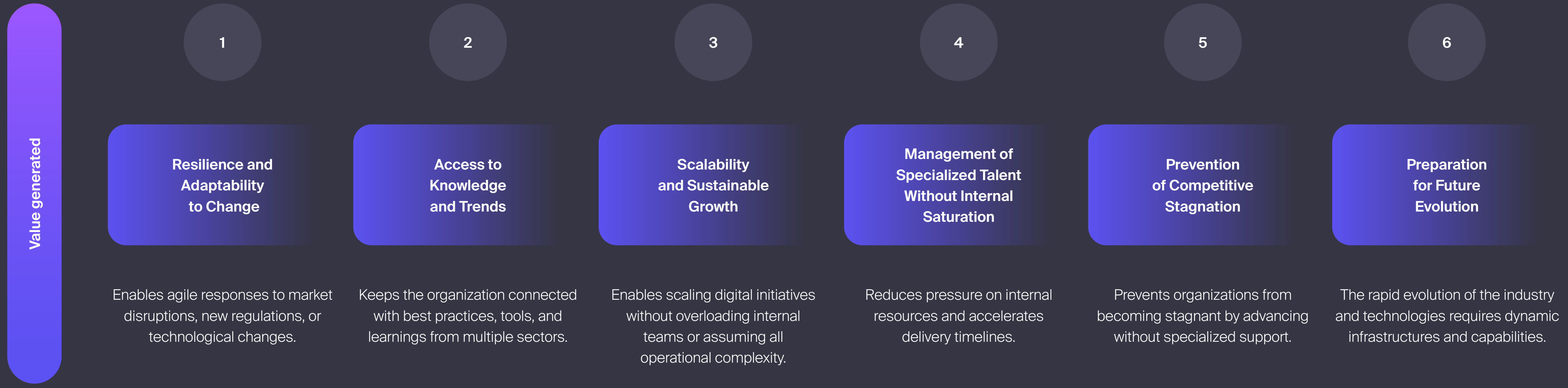
of companies struggle to find IT talent, making talent development a priority within strategic alliances.



more operational efficiency for companies that integrate AI through strategic partnerships.

The new era of code is led by shared vision, continuous collaboration, and a strategy based on co-creating value

Companies that commit to this type of alliance not only execute better; they think and decide with greater confidence and adapt more quickly. The benefits of having a strategic partner are multiple:



1 **Resilience and Adaptability to Change**

Enables agile responses to market disruptions, new regulations, or technological changes.

2 **Access to Knowledge and Trends**

Keeps the organization connected with best practices, tools, and learnings from multiple sectors.

3 **Scalability and Sustainable Growth**

Enables scaling digital initiatives without overloading internal teams or assuming all operational complexity.

4 **Management of Specialized Talent Without Internal Saturation**

Reduces pressure on internal resources and accelerates delivery timelines.

5 **Prevention of Competitive Stagnation**

Prevents organizations from becoming stagnant by advancing without specialized support.

6 **Preparation for Future Evolution**

The rapid evolution of the industry and technologies requires dynamic infrastructures and capabilities.

Ultimately, the new era of code not only changes the way code is created and developed — **it is a strategic opportunity for organizations to elevate their potential by relying on technology partners capable of leading, guiding, and executing the challenges of digital transformation.**

Softtek®

softtek.com