From Complexity to Control.

Transforming Modern Software Development.

April 2026

Crodox

Growing Complexity of Software Development

The demand for software is expanding rapidly as digitalization accelerates. While modern tools enable faster development than ever before, much of today's infrastructure was never designed for this pace or the vast scale of code being produced. This results in structural inefficiencies across software development workflows, increasing the likelihood of errors and ultimately slowing project delivery. Crodox addresses this challenge by reducing software complexity to its essentials: creating isolated, task-specific work environments where both developers and AI systems can operate efficiently, securely, and with full precision.

At its core, Crodox is built around three foundational pillars that define modern enterprise software development in the AI era:

- **Context-Aware AI:** providing accurate, bounded context for reliable AI-assisted development
- **Scalable Workflows:** enabling faster onboarding, parallel execution, and safe outsourcing
- **Secure Code:** limiting access, improving quality, and reducing cyberattack surfaces

Due to rapid technological advancements and the need for competitive edge in the digital economy, software has become indispensable for companies. It may have been sufficient to buy it "off the shelf" in earlier times. But nowadays almost every organization develops its own code or at least customizes standard code (e.g. SAP offerings). A key driver of this development is the increasing adoption of Low Code, No Code, Vibe Coding and other AI solutions. They make it easy to generate many lines of code in a very short time, while still requiring adherence to documentation, best practices, and governance for maintainability and security. The Github Copilot[1], for example, reduced the average time to open a pull request in 2025 from 9.6 to 2.4 days, speeding up the entire delivery cycle, according to their data statistics.

Yet this speed comes with a new bottleneck: without proper management, AI-generated changes can introduce errors, increase security exposure and accelerate technical debt. There is a large amount of legacy code that still needs to be maintained and managed. Surveys show that the share of the IT budget allocated to this challenge rose from 20%[2] on average to 40%[3] over the last years alone. As programmer Ward Cunningham[4] puts it:

---

[1] https://www.wearetenet.com/blog/github-copilot-usage-data-statistics
[2] https://www.mckinsey.com/capabilities/tech-and-ai/our-insights/tech-debt-reclaiming-tech-equity
[3] https://www.mckinsey.com/capabilities/tech-and-ai/our-insights/breaking-technical-debts-vicious-cycle-to-modernize-your-business
[4] https://www.youtube.com/watch?v=pqeJFYwnkjE

Shipping first-time code is like going into debt. A little debt speeds development so long as it is paid back promptly with refactoring. The danger occurs when the debt is not repaid. Every minute spent on code that is not quite right for the programming task of the moment counts as interest on that debt. Entire engineering organizations can be brought to a stand-still under the debt load of an unfactored implementation, object-oriented or otherwise.

With new concepts and technologies ahead, the amount of code that must be reviewed or maintained will increase much further. The key challenge is clear: to achieve sustainable growth, enterprises must modernize not only their code, but also the workflows, governance, and infrastructure surrounding software production.

Crodox provides the missing workflow layer that enables enterprises to develop software with AI-driven speed, while maintaining context, scalability, and security by design.

Source: Adobe Stock

The Challenges of Software Development

Even a functioning codebase must be regularly maintained, repaired, and optimized. Continuous software development is necessary. However, it is often akin to fixing a car while driving at high speed. The challenges are considerable. They include:

- **Growing Complexity of Legacy Systems.** Over time, legacy code accumulates across enterprises. Combined with evolving requirements and limited maintenance, historical code can become nearly unmanageable, slowing development and increasing risk. McKinsey reports that up to 70% of software used by the 500 largest U.S. companies was written more than 20 years ago[5]. Several systems are decades old: in 2024, Deutsche Bahn sought administrators familiar with Windows 3.11[6], while COBOL[7], over 60 years old, still powers critical banking and insurance applications.

  Modern software continuously evolves: every new feature or module adds interdependencies, increasing system size and complexity. This growth brings integration challenges, hidden flaws, and components that must be understood, tested, and optimized. Meta highlights key drivers of this rising complexity, including a focus on rapid delivery at the expense of long-term maintainability and developer churn, where new team members may make incompatible changes[8]. According to Meta, these factors create a "tangled web of hacks, workarounds, dead code, and unfinished tasks that ultimately make source code more difficult to maintain."

  Legacy code and evolving complexity make context-aware AI essential: without isolating code into meaningful, modular components, even AI-assisted tools risk producing errors that propagate across the system. Non-modular architectures make even minor feature updates highly risky[9].

  Maintaining legacy systems is expensive, often invisible to customers, and consumes a growing share of IT budgets. Surveys show enterprises now allocate approximately 40% of IT budgets to legacy maintenance. The U.S. federal government, for example, spends over $100 billion annually on IT, roughly 80% of it on legacy systems, with 51 of 138 critical systems

[5] https://www.mckinsey.com/capabilities/quantumblack/our-insights/ai-for-it-modernization-faster-cheaper-and-better

[6] https://www.chip.de/news/Verwunderliche-Anzeige-der-Deutschen-Bahn-Unternehmen-sucht-Admin-fuer-veraltetes-Windows_185125515.html

[7] https://www.sysmatch.com/new/role-of-cobol-in-banking/

[8] https://arxiv.org/html/2504.12517v2

[9] https://www.techolution.com/blog/5-reasons-postponing-legacy-modernization-2026-could-spell-catastrophic-risk/

deemed unsustainable yet essential for safety[10]. This technical debt limits innovation: budgets spent on maintaining the past could otherwise fund new features, AI deployment, and scalable development initiatives. McKinsey finds that companies actively addressing tech debt achieve revenue growth 20% higher than those who do not[11].

- **Security Risks.** Cybersecurity risks are rising rapidly. In 2025 alone, 70% of organizations reported being breached[12], with software vulnerabilities among the primary attack vectors[13]. The nonprofit organization Health-ISAC reported a 55% year-over-year surge in cyber incidents across the health sector in 2025, with trends pointing to continued escalation into 2026[14]. Legacy and highly complex systems are particularly exposed, creating significant security and operational risks. Within U.S. government systems, it is estimated that up to 70% of security vulnerabilities originate in legacy code[15]. Even without external attacks, system integrity can be compromised. Careless disclosure of source code to freelance developers or ungoverned AI tools introduces additional vulnerabilities and compliance risks. These realities underscore a fundamental principle: security must be embedded from the outset. Architecting systems as modular, well-defined components not only improves maintainability but also significantly reduces attack surface and systemic risk.
  The U.S. Department of Defense[16] therefore recommends decomposing monolithic systems into smaller, clearly defined components to reduce fragility and improve resilience.

- **Barriers to Sustainable Scaling.** Agile methods, low-code solutions, and AI accelerate software development but scaling safely is challenging:

[10] https://blog.sumaria.com/modernizing-legacy-systems
[11] https://www.mckinsey.com/capabilities/tech-and-ai/our-insights/demystifying-digital-dark-matter-a-new-standard-to-tame-technical-debt/
[12] https://qualysec.com/cybersecurity-statistics/
[13] https://www.weforum.org/publications/global-cybersecurity-outlook-2026/
[14] https://industrialcyber.co/reports/health-isac-reports-55-surge-in-cyber-incidents-in-2025-as-attacks-rise-and-escalation-looms-in-2026/
[15] https://blog.sumaria.com/modernizing-legacy-systems
[16] https://media.defense.gov/2025/Jun/23/2003742198/-1/-1/0/CSI_MEMORY_SAFE_LANGUAGES_REDUCING_VULNERABILITIES_IN_MODERN_SOFTWARE_DEVELOPMENT.PDF

- when key developers leave, projects can be delayed by months.
- Freelancers require repeated onboarding for complex codebases. Organizations are often hesitant to grant them full access to critical systems.
- Quick solutions increase technical debt, slowing future development. Suboptimal implementations make future work slower and more costly.

Scaling development sustainably requires controlled, modular approaches that enable rapid development while keeping systems maintainable, secure, and adaptable. Without this, organizations face rising costs, delayed projects, and compounding technical debt, limiting their ability to innovate effectively.

Enterprises need to move quickly while maintaining security and quality, a balance that existing tools cannot achieve.

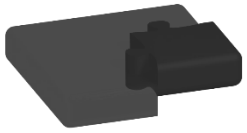
How Crodox Solves These Challenges

Crodox resolves these through its three pillars:

1. **Context-Aware AI:** precise environments for reliable AI-assisted development
2. **Scalable Workflows:** parallelized, safe, and fast development
3. **Secure Code:** structural security embedded at every step

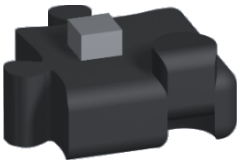
Crodox facilitates the automatic extraction of user defined components from an extensive source code into clear and easily manageable units. This way they can be edited separately and easily reintegrated. Here is how it works:

Step 1:



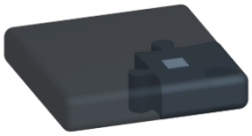
Crodox identifies and extracts all relevant components along with their dependencies from a complex codebase into a separate Git repository.

Step 2:



Developers or AI can work on isolated tasks in this standalone environment.

Step 3:



Once the changes are complete and validated, Crodox efficiently integrates them back into the main repository, minimizing conflicts.

By making software modular, Crodox enables both developers and AI to work independently and safely, reducing risks associated with legacy systems.

Why Existing Tools Cannot Solve It

Until now, no comprehensive solution existed to address these challenges. The rising share of IT budgets devoted to maintaining legacy code, increasing from 20% to an average of 40% over the past few years, underscores the growing impact of these unresolved problems.
Some tools can flag issues, but they cannot extract and reintegrate interdependent components. Some tools can extract and reintegrate interdependent components, but they operate only within predefined dependency structures, often requiring companies to upload their entire codebase to vendor systems - an approach that is typically unacceptable for enterprises. Other tools require feeding massive, noisy monoliths into analysis engines or AI prompts.

Context gets lost while tokens grow exponentially. Furthermore, due to token limits and lack of enterprise-specific patterns, AI coding assistants tend to hallucinate on huge repositories. This leads to insecure or even non-functioning changes that cause damage when they are integrated into the codebase. Crodox's unique extraction engine automates the safe decomposition of components at scale, completing in a very short time what would take manual teams months.

Source: Adobe Stock

Key Benefits of Working with Crodox

Crodox reduces the complexity of monolithic codebases by creating isolated, flexible, manageable and customized development environments. The main advantage is that no knowledge of the full codebase is needed. Developers or AI agents can concentrate on their specific tasks without having to worry about the details of the main codebase and its dependencies. Working with Crodox offers companies several benefits. The key ones are:

- **Faster Onboarding.** Crodox reduces onboarding time from months or even years to days.

- **Faster Scaling.** Crodox increases the flexibility in scaling: Different developers/AI can work independently on different tasks in parallel, increasing throughput and decreasing bottlenecks in peak times.

- **Increased Security.** With Cro-dox, there is no need to share the whole legacy code with freelancers, AI or other third parties. Apart from that, fewer employees have access to the full codebase, reducing the attack surface of the company. In addition, Crodox does not access or store the codebase.

- **Increased Code Quality.** Working on small, isolated slices of a large codebase with clear boundaries reduces bugs and improves maintainability. Code reviews – which are vital to improve the quality of a code – become more effective since reviewers do not have to know the whole codebase inside out. This enables more developers to take part in the review processes. It also facilitates to set consistent standards across teams of developers and reviewers. This way, technical debt gets reduced step by step.

- **Enterprise-Grade Foundation for AI at Scale.** Crodox makes AI adoption reliable in complex codebases through automated data labeling, isolated workbenches for controlled development, and reduced token footprints for faster, cheaper model performance.

Deep Dive

What makes Crodox unique is its ability to tailor complex workflows to a company's specific needs while also supporting predefined, standardized use cases. This is achieved by giving users the ability to shape and define their own templates. Each template consists of three core elements: the **Compiler**, which defines the relevant code objects and their dependencies. The **Environment**, which creates an isolated workbench around extracted components during production. And the **Implementation**, which automates the integration and referencing of those components within the workbench.

To begin the extraction process, the user selects the component he or she wants to modify in the source code. Crodox automatically presents a set of templates suited to extracting components from the selected file, allowing the user to choose the most appropriate option. Once a template is selected, its compiler generates a dependency graph that maps all related components and their relationships. The user then selects the component to be worked on, and Crodox automatically identifies all relevant code sections based on this dependency graph.

A dedicated Git branch is automatically created, after which a copy of the work environment is formed, and the identified code sections are transferred into it. This work environment operates independently from the original codebase. To enable seamless interaction with the extracted component, references are automatically generated so the component is fully integrated within the environment.

The implementation layer applies predefined rules based on the defined object types. This ensures that once Crodox completes the process, the work environment runs independently, contains all required code, and is fully integrated without any human intervention. The environment can then be shared with a developer along with a description of the required changes. Once the changes are completed, Crodox automatically detects them and inserts the updates back into the corresponding project branch in the codebase. This process enables a new approach to software production by allowing developers to contribute to any project without requiring deep insight into the surrounding codebase. Developers can remain focused on their area of expertise, while companies gain the ability to manage resources more efficiently and make better use of existing know-how.

As a result, production can scale more effectively to meet product demand, new projects can be formed faster, onboarding time is significantly reduced, and overall quality control is improved. Code reviews are no longer limited by deep codebase expertise, enabling broader participation and more consistent standards across teams.

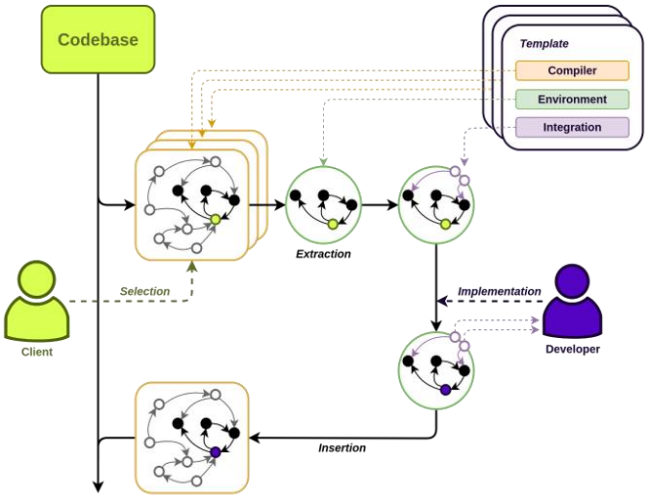


Fig. 1: How Crodox works, Source: Crodox Software GmbH.

What about AI?

One of the main challenges that companies face nowadays is keeping their rapidly growing codebase efficient, adaptable, and scalable. AI works much faster than humans, making it indispensable in software development. The trend is clear: the CEOs of Microsoft[17] and Google[18] state that around 30% of their code is already AI-generated, and Microsoft CTO Kevin Scott predicts this could rise to 95% in five years. However, effective AI use requires proper infrastructure and well-prepared data – something many enterprises struggle with. Surveys show that 78 to 90%[19] of companies have difficulties integrating AI with legacy systems, limiting scalability and business impact[20].

At enterprise-level, many face huge challenges:

- **Context Windows.** Growing token sizes from ever-expanding input data pose a major challenge for AI in software development. As repositories grow, it becomes increasingly difficult for LLMs (Large Language Models) to maintain context awareness. To avoid hallucinations, models require well-defined, bounded environments, yet legacy codebases often exceed even the largest context windows. While splitting repositories into semantic chunks can help, it risks missing critical cross-file dependencies or architectural relationships. This remains a fundamental, unsolved limitation of current AI tooling. The importance of this challenge is underscored by research at the Massachusetts Institute of Technology (MIT), which is exploring Recursive Language Models (RLMs) [21] that treat large codebases more like searchable databases rather than single-context inputs, offering a promising path toward better scalability and reliability.

- **Token Costs.** Deploying AI at enterprise scale can dramatically increase token costs, often far beyond the pilot phase. While an agent system may cost only a few dollars per day in testing, costs can multiply quickly in production due to repeated API calls and recursive context processing, reaching several thousand dollars per month. A documented case showed that a

[17] https://www.cnbc.com/2025/04/29/satya-nadella-says-as-much-as-30percent-of-microsoft-code-is-written-by-ai.html
[18] https://www.moneycontrol.com/technology/over-30-of-google-s-new-code-now-ai-generated-working-on-deeper-coding-experiences-sundar-pichai-article-13003845.html
[19] https://zapier.com/blog/ai-resistance-survey/ and https://www.zdnet.com/article/90-of-it-leaders-are-facing-difficulties-integrating-ai-with-other-systems-due-to-data-silos/
[20] https://www.techolution.com/blog/5-reasons-postponing-legacy-modernization-2026-could-spell-catastrophic-risk/
[21] https://arxiv.org/html/2512.24601v1

misconfiguration over 11 days led to $47,000 in token costs because two agents ran in a loop, constantly reloading the context[22]. Without cost dashboards or token limits, organizations can easily lose track. This phenomenon, referred to as the "invisible cost multiplier", highlights that effective AI use requires not only powerful models but also structured cost management and infrastructure planning.

- **Decrease of quality with increasing complexity.** AI-generated code often introduces subtle flaws that compound as system complexity grows. Studies show that the more complex the input code is, the more the performance degrades. Quality is even further deteriorating when tasks are complex and unstructured – but most legacy code is. Recent reports highlight the architectural issues – not model capability – as the root cause of these failures. While a pilot might show approximately 95% accuracy on curated data, production environments with real-world variability see accuracy fall to about 80%, and response latencies increase by over a factor of 20. This demonstrates how AI output reliability degrades without rigorous architectural design, automated testing, and monitoring frameworks[23].

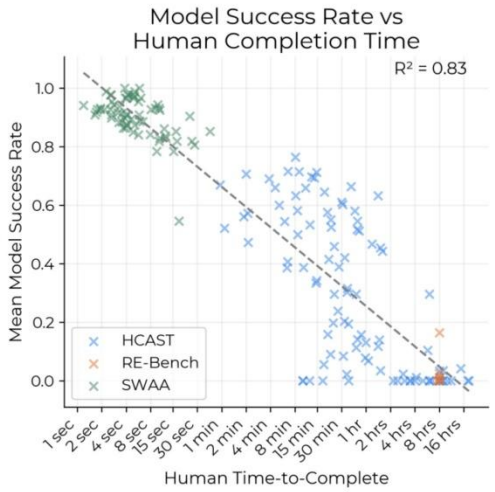


Fig. 2: Decrease of quality with increasing complexity. Source: Kwa et al, Measuring AI Ability to Complete Long Tasks.[24]

- **Limited Availability of Well-Structured Data.** LLMs rely on clean, well-labeled data to produce reliable outputs and reduce hallucinations. However, enterprise datasets are rarely systematically curated, and publicly available code data often lacks consistent labeling. Creating high-quality labeled data is expensive and time-consuming, and noisy input inevitably limits AI performance. As Donato Diorio, founder and CEO, puts it: "Without a systematic way to start and keep data clean, bad data will happen[25]." French AI scientist Yann LeCun highlights the scale of the challenge: a four-year-old

[22] https://t3n.de/news/ki-agenten-scheitern-nicht-am-modell-sondern-an-diesen-fuenf-architekturfehlern-1730278/
[23] https://t3n.de/news/ki-agenten-scheitern-nicht-am-modell-sondern-an-diesen-fuenf-architekturfehlern-1730278/
[24] https://arxiv.org/abs/2503.14499
[25] https://figarodigital.co.uk/articles/why-you-should-prioritise-data-cleanliness/?utm_source=chatgpt.com

child has already processed roughly 50 times more structured information than the largest LLMs[26]. This comparison underscores that, without substantial investment in high-quality input data, AI output quality will remain fundamentally constrained.

- **Necessity of Human Oversight and Validation.** AI is undeniably faster than humans – standard LLMs can generate 20 to 60 lines of code per minute – but the sheer volume of output creates a reliability challenge. As software engineer Addy Osmani calls it, the "70% problem" means that while AI can get roughly 70% of the way toward a solution quickly, the remaining 30% often requires disproportionate human effort to correct[27]. This is especially true for non-engineers, who lack the mental models to identify errors or understand architectural implications. Integration, maintainability, and safety still demand expert review. Moreover, AI cannot assume responsibility: LLMs are black boxes, providing little transparency about their reasoning or knowledge paths. Critical decisions – such as production code deployment – still require humans who can be held accountable. In short, AI serves as a powerful supplement in software development, but it cannot fully replace human judgment, expertise, or accountability.

Conclusion. You cannot run an express train on outdated tracks. Most AI deployments in enterprises fail not because of weak models, but because the infrastructure cannot support them. The consequences are clear: fragmented data silos that prevent unified model training, AI agents unable to act autonomously, security restrictions blocking access to critical datasets, and rapidly escalating token costs that make large-scale usage unsustainable[28]. Enabling AI effectively requires infrastructure designed for both scalability and cost efficiency. That is precisely what Crodox provides.

[26] https://www.youtube.com/watch?v=jmkTM2VSQoY
[27] https://addyo.substack.com/p/the-70-problem-hard-truths-about
[28] https://www.techolution.com/blog/5-reasons-postponing-legacy-modernization-2026-could-spell-catastrophic-risk/

From AI Potential to Productivity

Crodox is an AI-Enabler. Here is how:

- **Crodox optimizes AI efficiency and costs.** By automatically extracting only the minimal relevant slice of a codebase into isolated workbenches, Crodox removes the noise of entire monoliths that would overwhelm LLM context windows – no manual chunking required. This targeted approach not only maintains all critical links and dependencies (similar to MIT's Recursive Language Models), but also reduces the input size for AI models, leading to faster processing, shorter response times, and lower computing effort. Less input also means fewer output tokens, directly cutting token-related costs. Initial estimates from Crodox suggest that isolating workbenches can reduce token usage by at least a factor of 10, making large-scale AI deployment more practical and economical.

- **Crodox handles complexity.** By structuring complex codebases and cutting them into independently editable slices, it makes working on tasks easy without endangering the day-to-day business. It offers a secure way to put AI into good use to reduce technical debt.

- **Crodox generates high-quality labeled data**. By slicing the codebase into task-specific environments, Crodox captures requests and interactions within the context of each environment, effectively turning that context into its own label. Every review produces structured data that trains AI agents to deliver smarter, more accurate suggestions. The results are domain-specific agents fine-tuned on labeled data that reflects local code structure, guidelines, and protocols – making AI-driven development both reliable and secure. Unlike many AI tools that rely on unstructured or unlabeled data, Crodox ensures high-quality training data without the time and cost of manual labeling.

- **Crodox enables seamless human-AI collaboration.** By providing precisely scoped, clean code environments, Crodox creates the foundation for effective human-AI interaction. Each project – such as a frontend development – helps AI agents become more accurate for that specific component. Developers can focus on reviewing, maintaining, and making creative decisions, while AI handles routine coding within well-defined boundaries, accelerating development without compromising quality.

The true power of AI emerges when advanced models are paired with high-quality labeled data and supported by infrastructure that enables sustainable, scalable use. Crodox provides this.

Where Crodox Stands Today

Nowadays, software development presents companies with problems that are almost impossible to solve. When ever-increasing speed, complex codebases, and limited resources collide, quality problems are bound to arise sooner or later. The safe and efficient use of AI is a solution, but it requires a working environment that is currently rarely found in companies. Crodox solves this. And we are ready to go – with you:

- Our team spent the last three years developing this product, from an initial proof of concept to the practical implementation.
- The first pilot clients are using Crodox productively, putting it into effective use while giving us useful feedback for further features.
- Crodox can start distributing its software to clients, offering various usage models tailored to the specific needs of individual customers.
- Companies can continue to use their existing legacy code, no matter which code language it was written in.
- Crodox works with isolated workbench environments, so companies do not have to change their IT infrastructure. By that, it also offers flexibility: One sole developer can benefit from the tool as well as whole teams.
- Crodox neither accesses nor stores a company's code, eliminating potential compliance concerns

Possible Use Cases for Crodox are:

- **Isolated feature development:**
  Planned features can be developed and tested independently while remaining fully integrated with the broader codebase.
- **Simplified debugging:**
  Errors can be isolated along with their dependencies, enabling faster and more precise bug fixing.
- **Modernization & technical debt reduction:**
  Legacy modules can be incrementally refactored or rebuilt in isolated environments, improving maintainability and reducing technical debt without disrupting the system.
- **Flexible resource allocation:**
  Individual tasks can be assigned to external developers and automatically reintegrated.
- **Parallel development:**
  Codebases can be divided into components, enabling multiple teams to work simultaneously and accelerate delivery.
- **Isolated QA and testing:**
  Code changes can be tested in dedicated, isolated environments, enabling faster and more reliable quality assurance without impacting the broader system.

- **Reliable AI-assisted development:**
  AI outputs can be improved by operating in isolated, context-specific environments that generate structured, high-quality labeled data, while human review ensures accountability, seamless integration, and adherence to guidelines.

- **Cost-efficient AI scaling:**
  Token usage and processing overhead can be reduced by limiting AI context to relevant code segments, enabling scalable AI deployment without excessive costs.

There is even more coming to Crodox: We are building a platform where companies can share tasks as isolated workspaces, enabling them to collaborate securely with external experts. Features such as a task marketplace, bidding process, expertise matching, and secure code sharing go far beyond traditional code hosting or project management tools. Unleash your developers and AI agents by giving them dedicated workbenches you and they can rely on.

You are a company that develops its own code?

You have a significant and expanding codebase?

Then Crodox can be of enormous value to you.

If you want to put us to the test, please contact us:

info@crodox.com

www.crodox.com