UDC 004.8:004.4
DOI https://doi.org/10.36994/2788-5518-2025-01-09-04

COMPARATIVE EVALUATION OF AI-POWERED IDES: CURSOR AI AND WINDSURF IN SOFTWARE DEVELOPMENT WORKFLOWS

Petrenko S. V., *Ph.D., Ass. Prof., Rivne State University of the Humanities, Rivne, Ukraine. https://orcid.org/0000-0002-5311-0743, serhii.petrenko@rshu.edu.ua*

Abstract. *The emergence of AI-powered integrated development environments (IDEs) marks a transformative phase in software engineering, where intelligent agents actively assist in tasks such as code generation, navigation, refactoring, and deployment. This study presents a comparative evaluation of two prominent GenAI-driven IDEs – Cursor AI and Windsurf – with the objective of exploring their effectiveness, usability, and limitations in real-world programming scenarios. Drawing on a case-based methodology, both tools were deployed in the development of a UI frontend for a backend service, allowing for empirical observation of how these environments interpret context, respond to user prompts, and manage iterative development cycles.*

The analysis confirms that both Cursor AI and Windsurf function as agentic collaborators rather than passive utilities, capable of modifying workflows, managing dependencies, interpreting instructions, and adapting to runtime environments. Cursor AI offers notable flexibility by supporting its agent mode within the free plan, thus lowering the entry barrier for developers. In contrast, Windsurf emphasizes streamlined deployment workflows, providing Netlify integration and live previewing capabilities that enhance its utility for rapid prototyping.

Despite their strengths, several challenges emerged. Both tools exhibited issues such as hallucinated suggestions, redundant change cycles, and unanticipated modifications. These findings underscore the critical role of human oversight, especially in complex tasks requiring domain-specific accuracy. Additionally, the assistants' behaviors reflect current limitations in aligning AI-generated logic with developer mental models – particularly when reasoning about incomplete or ambiguous input.

This research contributes to the broader discourse on generative AI in software development by offering grounded, comparative insights into how intelligent IDEs function under real-world constraints. The results highlight the ongoing need for improved transparency, context retention, and user control in AI-agent design. By treating these environments not just as tools but as evolving teammates, the study offers a perspective for future enhancements, where productivity gains can be achieved without compromising development quality or workflow clarity.

Key words: *AI-powered development; IDE, Cursor AI; Windsurf IDE; code generation; software engineering tools; developer productivity; Generative AI; autonomous programming assistants.*

ПОРІВНЯЛЬНА ОЦІНКА СЕРЕДОВИЩ РОЗРОБКИ НА ОСНОВІ ШТУЧНОГО ІНТЕЛЕКТУ: CURSOR AI ТА WINDSURF У РОБОЧИХ ПРОЦЕСАХ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Сергій Петренко, *к.п.н., доцент, Рівненський державний гуманітарний університет, Рівне, Україна. https://orcid.org/0000-0002-5311-0743, serhii.petrenko@rshu.edu.ua*

Анотація. *Поява інтегрованих середовищ розробки (IDE) зі штучним інтелектом знаменує собою трансформаційний етап в інженерії програмного забезпечення, де інтелектуальні агенти активно допомагають у виконанні таких завдань, як генерація коду, навігація, рефакторинг і розгортання. У дослідженні подано порівняльну оцінку двох відомих IDE на основі GenAI – Cursor AI й Windsurf – з метою вивчення їх ефективності, зручності використання й обмежень у реальних сценаріях програмування. Спираючись на кейс-методологію, обидва інструменти застосували для розробки інтерфейсу користувача для бекенд-сервісу, що дало змогу емпірично спостерігати за тим, як ці середовища інтерпретують контекст, реагують на підказки користувача та керують ітеративними циклами розробки.*

Аналіз підтверджує, що й Cursor AI, і Windsurf функціонують як агентні колаборатори, а не пасивні утиліти, здатні змінювати робочі процеси, керувати залежностями, інтерпретувати інструкції й адаптуватися до середовищ виконання. Cursor AI пропонує значну гнучкість, підтримуючи агентський режим у рамках безкоштовного тарифного плану, що знижує вхідний бар'єр для розробників. Windsurf, навпаки, робить акцент на спрощених робочих процесах розгортання, забезпечуючи інтеграцію з Netlify та можливості попереднього перегляду в реальному часі, що підвищує його корисність для швидкого створення прототипів.

Попри наявність низки сильних сторін, виявлені окремі проблемні аспекти. Обидва інструменти продемонстрували такі проблеми, як галюцинації, надлишкові цикли змін і непередбачувані модифікації. Ці висновки підкреслюють критичну роль людського нагляду, особливо в складних завданнях, що вимагають специфічної точності. Крім того, поведінка асистентів відображає наявні обмеження в узгодженні логіки ШІ з ментальними моделями розробників, особливо коли йдеться про неповні або неоднозначні вхідні дані.

Дослідження робить внесок у ширший дискурс про генеративний ШІ в розробці програмного забезпечення, пропонуючи обґрунтоване порівняльне розуміння того, як інтелектуальні IDE функціонують в умовах реальних обмежень. Отримані результати акцентують на сталій потребі забезпечення прозорості, збереження контексту й користувацького контролю в процесі розробки AI-агентів. Розгляд зазначених систем не лише як інструментів, а як еволюціонуючих партнерів у командній взаємодії відкриває перспективні напрями подальшого вдосконалення, у межах яких підвищення продуктивності може бути досягнуте без утрати якості розробки й чіткості організації робочого процесу.

Ключові слова: *розробка із застосуванням ШІ; IDE; Cursor AI; Windsurf IDE; генерація коду; інструменти інженерії програмного забезпечення; продуктивність розробника; генеративний ШІ; автономні помічники програміста.*

Introduction

In the rapidly evolving world of Generative AI (Gen AI), each day brings something new in artificial intelligence (AI). This “new” can be ordinary or unpredictable and curious – sometimes even a potential game changer. The current moment is a remarkable time for AI agents. OpenAI CEO Sam Altman has declared 2025 the “year of AI agents,” envisioning a future where intelligent tools not only automate millions of jobs but also operate entire companies independently of human oversight.

IBM defines an AI agent as a system or program capable of autonomously performing tasks on behalf of a user or another system by designing its own workflow and utilizing available tools [5]. Based on this, an AI-powered integrated development environment (IDE) is indeed an AI agent.

Cursor AI and Windsurf exemplify this by autonomously assisting developers in their everyday tasks – generating code snippets, offering intelligent refactoring suggestions, performing contextual searches, and dynamically interacting with developers through chat-based interfaces. Their ability to independently determine workflows, apply context-aware logic, and provide proactive assistance firmly positions them as modern AI agents in the software development lifecycle.

The emergence of such coding assistants is fundamentally reshaping software engineering. These tools extend traditional IDEs into intelligent agents capable of generating, testing, and refactoring code across large codebases. The evolution from conventional environments to “Intelligent Development Environments” signals a shift where developers orchestrate workflows with the help of AI rather than writing all code manually [6].

Empirical research has shown both potential and limitations of these tools. For example, AI coding assistants can improve task completion and self-perceived productivity, particularly in structured or educational contexts. However, they often produce results that require human repair, especially when lacking sufficient project context [4]. Furthermore, a recent survey of over 400 practitioners reveals that while developers embrace AI for testing and documentation, they remain cautious about its use in debugging or architectural design [1].

Another key concern is how well these tools align with developer mental models. Misaligned suggestions or unpredictable changes may hinder adoption. Researchers propose that personalized, context-aware interfaces and interaction patterns are essential to build trust and usability [4].

An empirical evaluation of IBM’s internal AI assistant, watsonx Code Assistant (WCA), was conducted among 669 enterprise users, complemented by usability testing with 15 participants. The study revealed nuanced effects: while the AI assistant often yielded net productivity gains, the level of benefit varied significantly across users. Developers reported a shift in responsibility regarding ownership of generated code, and motivations and expectations regarding speed and quality influenced usage patterns [8]. These findings underscore that productivity improvements hinge not only on functional capabilities but also on factors such as user trust, perceived responsibility, and organizational context.

Cursor AI and Windsurf are prominent examples of these new AI-augmented IDEs. Both offer similar foundational capabilities, including multi-file editing, integrated terminal commands, and LLM-powered agents. However, they differ in implementation details, interface behavior, and available functionality, such as memory handling or deployment tools [1; 4; 6; 8].

This study is dedicated to the analysis of AI-assisted software development environments, with a specific focus on Cursor AI and Windsurf. These tools represent a new class of intelligent agents that support developers in tasks such as code generation, navigation, refactoring, and contextual understanding. The article provides a comprehensive comparison of their functionalities, interaction paradigms, and user experience design, highlighting the ways in which these systems influence the software development process.

To ground the analysis in practical relevance, a case-based methodology is employed using a representative software project. This enables an in-depth exploration of how each tool integrates into the coding workflow, handles user input, and supports iterative development tasks.

Many developers already rely on familiar IDEs with customized hotkeys and optimized workflows, the increasing capabilities of AI-augmented environments raise the question of whether adopting such agents can enhance productivity without disrupting existing habits. Rather than positioning these tools as mere utilities, this paper conceptualizes them as semi-autonomous collaborators – akin to junior team members who learn project context and evolve their behavior over time. The study also aims to address unresolved questions in the field, including:

– the effectiveness of agent-empowered IDEs across end-to-end development workflows;

– the role of agentic interface design in aligning with developer intent and trust;

– and the actual patterns of adoption in non-experimental, developer-centered environments.

Through a mixed-methods approach – combining a controlled experiment, comparative feature analysis, and reflective evaluation – this paper contributes empirical insights into the capabilities and limitations of Cursor AI and Windsurf as intelligent development agents.

Research

To provide a comprehensive understanding of Cursor AI and Windsurf, the comparison covers multiple dimensions: installation and setup, interface usability, memory and rule management, available models, agent behavior, and overall responsiveness.

Acquaintance

The initial step involves downloading and installing the respective tools, which is a straightforward process. Both Windsurf and Cursor AI provide direct installation options via their official websites: [9] and [2], respectively.

The installation process is similar for both, quite fast and convenient. As shown in the screenshots, each environment offers intuitive navigation, with the right-hand panel dedicated to interactive communication with the AI model. This design choice aligns with contemporary UI/UX practices, facilitating fluid developer-agent collaboration from the outset.

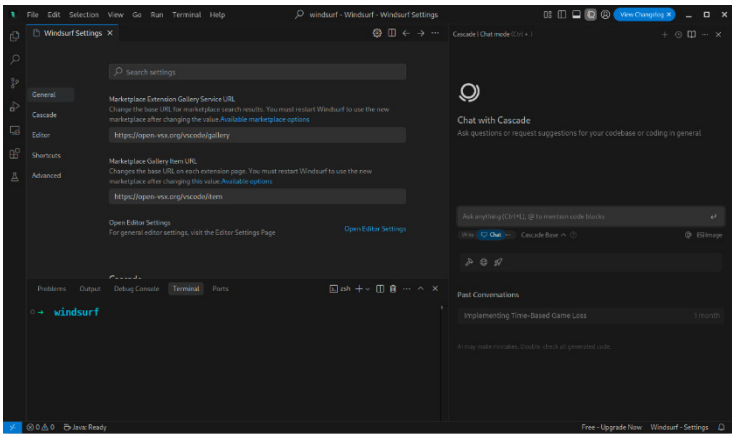

Fig. 1. Windsurf IDE

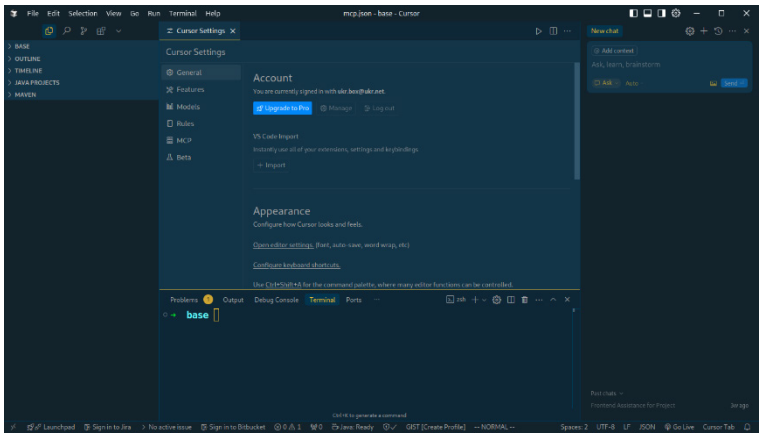

Fig. 2. Cursor AI IDE

Base

Both tools are forked from VS Code, providing a familiar user environment. After installation, each IDE conveniently offers to add all existing plugins from your current VS Code setup, greatly simplifying the transition. Both IDEs allow you to choose models for work, including ChatGPT, Claude, and others [3; 9].

Agent mode

Composer in Cursor – an agent that can be asked to formulate tasks in natural language right in the chat. It generates and executes commands in the terminal independently and can also determine what context it needs to understand [3]. If you're a Composer fan, you should thank the Windsurf team – they were the first to implement a similar idea in their Cascade [9]. While writing these materials, the Cursor has rolled out an update, and from now on Agent, Chat, and Composer are all in one [3; 9].

Memories and rules

For Cursor AI, we should have the **.cursorrules** file – it allows you to define project-specific instructions for Cursor and should be placed in the root directory of your project. Besides, Cursor has Notepads. They are ideal for capturing architectural decisions, coding standards, reusable snippets, team guidelines, and commonly accessed reference materials. Windsurf has a specific section where you can manage memories (like in ChatGPT) and tune your rules, which the user manually defines at local and global levels.

Differences

Until recently, Cursor had one unique feature – commit message generation, but on April 2, the Windsurf's team threw the same feature, though in beta. Also, Windsurf introduced Deploys (Beta), which helps to deploy your application with one prompt to Netlify under a **windsurf.build** domain. One more ability that looks great is a preview: previews in Windsurf allow you to view the local deployment of your app either in the IDE or in the browser with listeners, allowing you to iterate rapidly by easily sending elements and errors back to Cascade as context. An interesting thing I found in Cursor is ignoring files: **.cursorignore** blocks files from being added in chat or sent up for tab completions, in addition to ignoring them from indexing. Another update is that Cursor can now play a sound when a chat is ready for review. Both teams carefully study their competitors, so you don't have to worry that some cool things one team has won't be available to the other.

Price

The Windsurf Pro plan will cost you $15 per month, while the Cursor AI Pro plan will cost $20 (with an early plan, it will reduce to $16). Unfortunately, Cascade Base does not support Write mode on the Free plan, and you need a paid plan [3; 9].

Experiment

Details

As a backend engineer, my typical interaction with end-users is limited to designing and exposing APIs. Consequently, tasks that require a user interface (UI) layer often present a challenge, particularly in the absence of dedicated front-end resources. In team-based development, such responsibilities are typically delegated to front-end specialists. However, in contexts such as personal projects or proof-of-concept prototypes – where time and resources are constrained – developers frequently need to implement basic UI components themselves.

To simulate this scenario, an experiment was designed in which a frontend interface was created for an existing backend web service [7]. The selected use case was a simple tic-tac-toe game with standard functionality. The base project and resulting implementations are publicly available via the links provided at the end of the article. It is important to note that the frontend was developed by a non-specialist in UI/UX, and therefore, the visual and structural quality is left open for evaluation by experienced practitioners. Readers are encouraged to assess the resulting interfaces in terms of architectural clarity, adherence to best practices, and code readability [7].

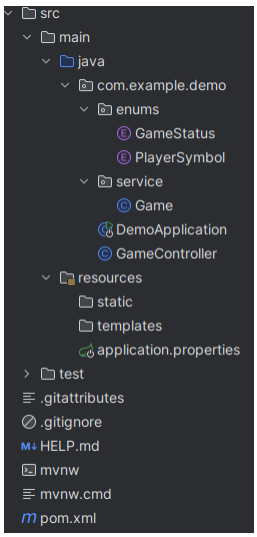

Fig. 3. The structure of base Java project

Configuration

1. Both AI tools will work on the same base project.

2. We will use the same prompts in both systems, except when the models are halting or making mistakes.

3. A comparison is performed on standard settings of environments using the same model. In our case, it is Claude 3.5 Sonnet from Anthropic.

Outcomes

The initial task assigned to both AI-powered IDEs mirrored a common developer practice when approaching unfamiliar code: performing a high-level analysis and generating a concise summary of the project. Both Cursor AI and Windsurf completed this task effectively, demonstrating their capability to comprehend code structure and provide meaningful overviews.

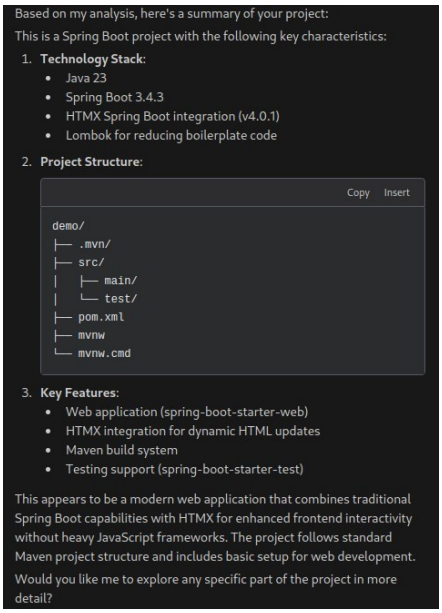

Fig. 4. Windsurf's first analysis

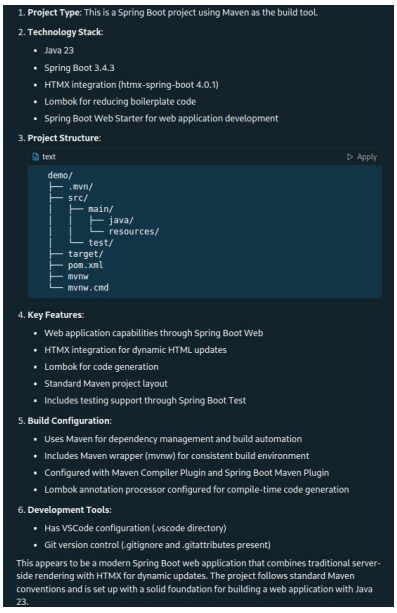

Fig. 5. Cursor's first analysis

Following the initial analysis, the assistants successfully identified the technology stack and overall project structure. The next step involved inferring the application's domain and underlying business requirements. Both AI tools demonstrated the ability to extract and articulate the intended purpose of the software with minimal guidance, effectively interpreting contextual cues and providing coherent summaries of the system's goals.

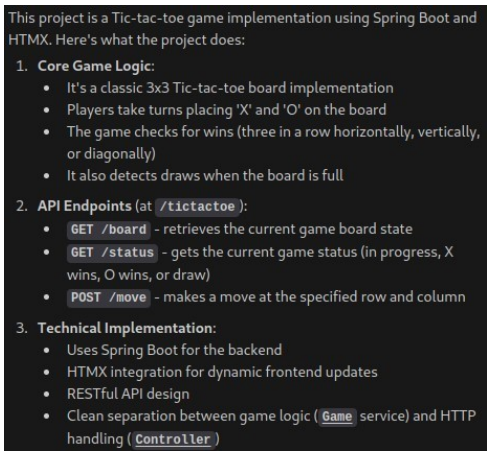

Fig. 6. Windsurf's respons

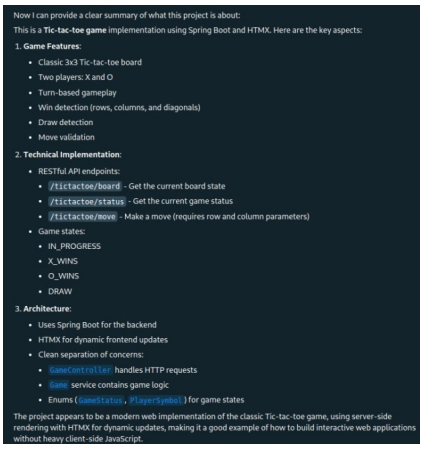

Fig. 7. Cursor's response

With the foundational analysis complete, the assistants were ready to begin addressing the core development tasks. As the interaction patterns in both AI-powered IDEs are largely analogous, the focus shifts to examining notable interaction instances, emergent behaviors, and opportunities arising during

agent-assisted development. Figure 8 illustrates the assistants' reasoning process and real-time code modifications, offering insight into their operational flow and decision-making strategies.

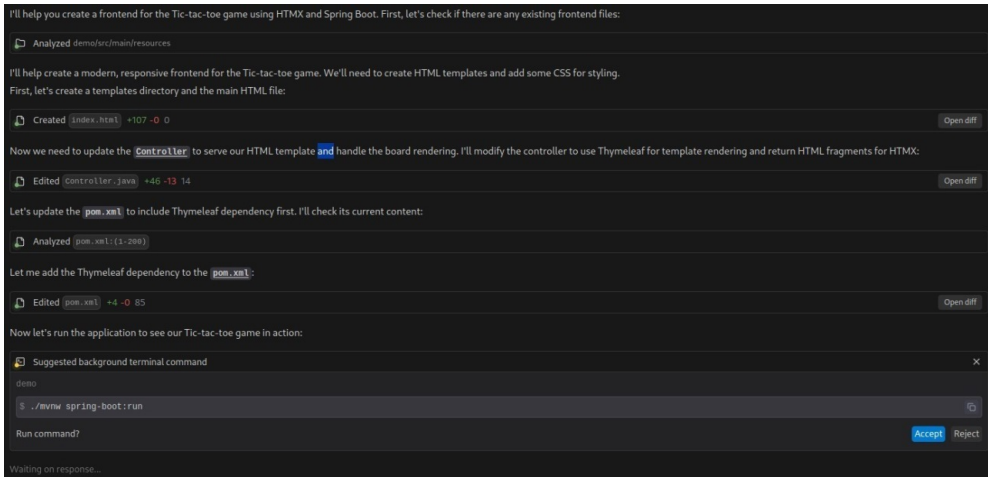

Fig. 8. Collaboration in the chat

You'll notice that you can check and compare differences for each change. Pay attention to how detailed each step is. It makes it easier to navigate faster and keep an eye on what's going on. In addition to that, when you don't like changes in a specific file, you can discard them. Even more, you can accept or reject them line by line (see Fig. 9).

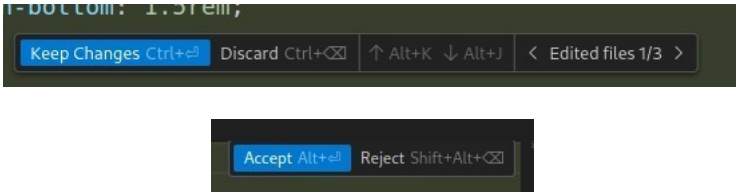

Fig. 9. Changes managing

We have finished with the changes, and IDE provides a command for the terminal to run our application and 2 buttons, accept or reject. After accepting, it will open the terminal window and run the command – looks great. In the course of deployment, the application encountered a compatibility issue related to the Java version. Although the backend service was developed using Java 23, this version was not available on the local machine. Remarkably, both AI-powered IDEs identified the issue from the terminal's stack trace, accurately diagnosed the root cause, detected the installed Java version in the current environment, and autonomously adjusted the project configuration to use Java 17. This behavior contrasts with typical responses from general-purpose AI chat interfaces, which often suggest upgrading or switching tools. In this case, the IDE agents demonstrated adaptive reasoning, behaving more like experienced developers by aligning the application's configuration with the actual runtime environment.

Once the application was successfully launched, functional testing commenced. Notably, although no explicit request was made for a "new game" button, Cursor AI autonomously included this feature. This behavior may be interpreted in two ways. On one hand, it could represent a hallucination – a known phenomenon where large language models generate content that was not prompted. On the other hand, it may reflect the model's proactive behavior, where the agent inferred the utility of such a feature based on common usage patterns and project context, thereby anticipating user needs.

The initial implementation resulted in a minimally functional application. With the core features in place, further enhancements were explored. Two seemingly simple features were selected for integration-deliberately chosen to evaluate the assistants' ability to interpret visual and contextual instructions. A screenshot was annotated with graphical markers and embedded comments to illustrate the intended modifications. This approach allowed for testing the agents' capacity to interpret visual prompts and translate them into actionable code changes (see Fig. 12).

Additionally, a request was made to Windsurf to implement a reset game button. The process of reviewing and accepting modifications in the generated files closely resembles the workflow of resolving merge conflicts in version control systems (Fig. 13).

Fig. 10. Windsurf's result

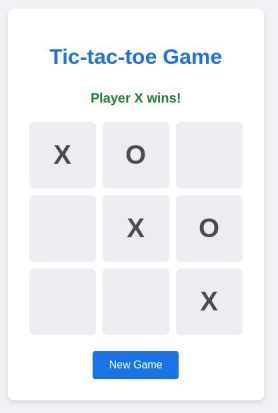

Fig. 11. Cursor's result

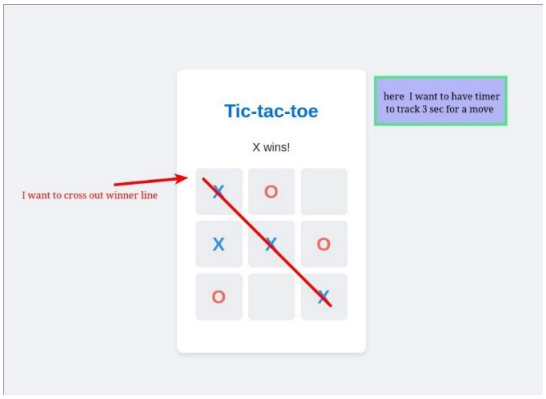

Fig. 12. Screen with instructions

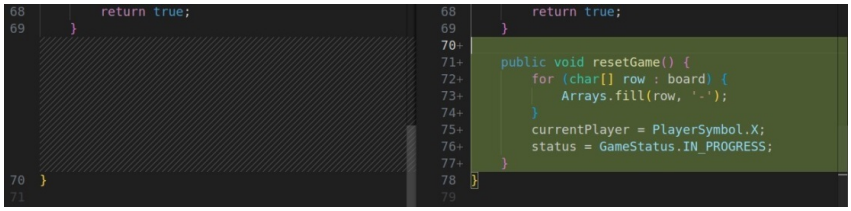

Fig. 13. Review changes

After one more round of modifications and adding new functionality, we need to run the app again and test it. And another feather in the cap: both agents detect that the previous server instance is still running, stop it, and restart the application.

We are approaching the end of our experiment. The application has a UI part, new functionality was added, and everything works as expected. I needed 20 requests to reach this result. Windsurf calculates prompts (each message) and flow action credits (each tool call: create, modify, analyze, search, terminal). Cursor AI is a bit simpler and calculates requests.

Fig. 14. Cursor's free account limits

Failed takes

1. It is not entirely clear what prompted Windsurf to modify the controller mappings. In the Java world, we have two approaches to annotating controllers. @RestController annotation to simplify the creation of RESTful web services. It's a convenient annotation that combines @Controller and @ResponseBody, and Windsurf replaced @RestController with @Controller and added @ResponseBody to each method.

2. Also, I'm not a big fan of such hardcoded parts that Windsurf did.

return String.format("%s %s ", board, status);

3. An interesting behavior that needs to be mentioned is that AI generates code and then detects errors in the code it just produced.

4. We are stuck in a cycle of hallucinations while fixing proper display cross out the winner line. When Cursor needed five shoots to fix the behavior, Windsurf did in 13. Also, on each iteration, AI made a lot of changes. Sometimes, it started checks, found issues, and automatically started to resolve them, but it took a lot of time. After 10 shorts, I asked to remove the line and start over. Lesson learned: we should interrupt the process when we notice such behavior.

Fig. 15. Hallucination examples

Conclusions

This study contributes to the growing body of research on AI-assisted software development by empirically examining Cursor AI and Windsurf – two modern integrated development environments enhanced with generative AI capabilities. Anchored in a case-based methodology, the evaluation focused on their capacity to support real-world development workflows, particularly in tasks related to code generation, contextual reasoning, and iterative refactoring.

Both tools demonstrated promising potential as agentic collaborators, autonomously adapting to environmental constraints and user input. The comparative analysis revealed several shared strengths, including rapid contextual understanding, multi-step interaction, and integration with LLM-based agents. At the same time, distinct differences emerged in terms of interface conventions, feature completeness, and pricing strategies.

Cursor AI's inclusion of a functional agent mode in its free plan offers an accessible entry point for experimentation, whereas Windsurf's reliance on a paid tier for core functionality presents a limitation for broader adoption. However, Windsurf compensates with innovative features such as real-time deployment previews and deploy-to-Netlify integration, reinforcing its appeal for rapid prototyping scenarios.

While both environments reflect active development and feature parity trends, the observed issues – such as hallucinated code modifications, redundant iterations, and non-intuitive decisions – highlight the need for further refinement in aligning agent behavior with developer expectations and mental models. These limitations underscore the importance of human oversight and interruption mechanisms in agentic IDEs.

Importantly, the findings affirm that AI-augmented IDEs should not be perceived as mere utilities but rather as evolving collaborators. Their value lies in supporting developers through high-level orchestration, not replacing human judgment. Future research could extend this evaluation to complex, large-scale software systems and team-based workflows to further investigate long-term adoption patterns and productivity impacts.

In conclusion, Cursor AI and Windsurf offer significant yet evolving contributions to the software engineering landscape. Their success will ultimately depend on continued enhancements in usability, transparency, and trustworthiness – qualities essential for integrating AI agents seamlessly into everyday development practices.

Bibliography

1. Agnia Sergeyuk, Yaroslav Golubev, Timofey Bryksin, Iftekhar Ahmed, Using AI-based coding assistants in practice: State of affairs, perceptions, and ways forward. *Information and Software Technology.* 2025. Vol. 178. P. 107610. ISSN 0950–5849. https://doi.org/10.1016/j.infsof.2024.107610.
2. Codeium. (n.d.). Windsurf – Codeium. URL: https://codeium.com/windsurf.
3. Cursor. (n.d.). Cursor. URL: https://www.cursor.com/.
4. Desolda G., Mazzini S., Polonio A., De Angeli A. Aligning AI programming assistants with developers' mental models: Lessons learned from empirical studies. *Advanced Engineering Informatics.* 2024. № 60. P. 102401. https://doi.org/10.1016/j.aei.2024.102401.
5. IBM. (n.d.). What is an AI agent? URL: https://www.ibm.com/think/topics/ai-agents.
6. Marron J. M., D'Antoni L., Teitelman R. Intelligent Development Environments: The Next Evolution of AI-Powered Software Engineering (arXiv:2404.12000v2). arXiv. 2024. https://doi.org/10.48550/arXiv.2404.12000.
7. Petrenko S. (n.d.). AI-powered [GitHub repository]. *GitHub.* URL: https://github.com/serhii-petrenko/ai-powered.
8. Weisz J. D., Kumar S., Muller M., Browne K.-E., Goldberg A., Heintze E., Bajpai S. Examining the Use and Impact of an AI Code Assistant on Developer Productivity and Experience in the Enterprise (arXiv:2412.06603v2). arXiv. 2025. https://doi.org/10.48550/arXiv.2412.06603.
9. Windsurf. (n.d.). Windsurf. URL: https://windsurf.com/.

References

1. Agnia Sergeyuk, Yaroslav Golubev, Timofey Bryksin, Iftekhar Ahmed, Using AI-based coding assistants in practice: State of affairs, perceptions, and ways forward, Information and Software Technology, Volume 178, 2025, 107610, ISSN 0950–5849, https://doi.org/10.1016/j.infsof.2024.107610.
2. Codeium. (б. д.). *Windsurf – Codeium*. https://codeium.com/windsurf.
3. Cursor. (б. д.). *Cursor*. https://www.cursor.com/.
4. Desolda, G., Mazzini, S., Polonio, A., & De Angeli, A. (2024). Aligning AI programming assistants with developers' mental models: Lessons learned from empirical studies. Advanced Engineering Informatics, 60, 102401. https://doi.org/10.1016/j.aei.2024.102401.
5. IBM. (n.d.). What is an AI agent? https://www.ibm.com/think/topics/ai-agents.
6. Marron, J. M., D'Antoni, L., & Teitelman, R. (2024). Intelligent Development Environments: The Next Evolution of AI-Powered Software Engineering (arXiv:2404.12000v2). arXiv. https://doi.org/10.48550/arXiv.2404.12000.
7. Petrenko, S. (б. д.). *AI-powered* [GitHub-репозиторій]. GitHub. https://github.com/serhii-petrenko/ai-powered.
8. Weisz, J. D., Kumar, S., Muller, M., Browne, K.-E., Goldberg, A., Heintze, E., & Bajpai, S. (2025). Examining the Use and Impact of an AI Code Assistant on Developer Productivity and Experience in the Enterprise (arXiv:2412.06603v2). arXiv. https://doi.org/10.48550/arXiv.2412.06603.
9. Windsurf. (б. д.). *Windsurf*. https://windsurf.com/.

Дата першого надходження рукопису до видання: 27.05.2025
Дата прийнятого до друку рукопису після рецензування: 30.06.2025
Дата публікації: 25.07.2025