58 ·
October 2025
COLUMN

The wave of massive change called AI was shaking every corner of our lives. This was not simply the emergence of new technology, but a flow that changed the way of thinking, working, and even the structure of society as a whole. For the past eight months, I have been learning and experimenting every day in this flow of change, continuing my own journey. During this time, I came to see AI not just as a simple tool, but as a companion that constantly stimulates myself through work, creation, learning, and all aspects of life. Rather than blindly trusting AI, I have honed my own 'killer move' through an attitude of keeping a careful distance and actively accepting it at the same time.

My learning method was somewhat unique, with 70% learned visually and 30% learned through hands-on experience. The reason I accepted this ratio was because my experience was not a developer's life. Although it was advertised that non-developers could develop through 'vibe coding', I understood that there were actual limitations. I was able to create and modify a company homepage with Cursor AI, and try a MBTI determination program with the Replit program through vibe coding.

However, as a PM who has experienced building PLM in a company, there were limitations to non-developers making programs. It is welcome to make it as a hobby, but after experiencing many issues that occur after the program is launched, I learned my own learning formula that collaboration with developers is more efficient. The knowledge gained from lectures, books, and studies became the foundation, and practice and trial and error connected that knowledge to reality.

While listening to the lecture of CEO Lee Boo-il and following the Python code using ChatGPT, I first felt that AI was not a simple language model but a powerful practical tool. I remember vividly that I followed well on the first day, but on the second day, my laptop battery ran out and I was disappointed, but even this experience became part of the learning process. I realized that AI learning is not only about storing knowledge in the head, but also about accepting it in life and environment. Failures and happenings

also became assets, and one by one, coordinates were marked on my learning map. What was important was not speed, but the process of constantly learning, recording, and reusing was much more valuable.

Yonghyo Ryu
He is an executive at Diwon and is active as a leader of the Facebook group 'Concept Map Research Institute'. He helps organize corporate processes with consulting and design thinking from a business perspective, and is interested in creating concept maps with one big size.
Email Yonghyo.ryu@gmail.com
Blog PLMIs.tistory.com

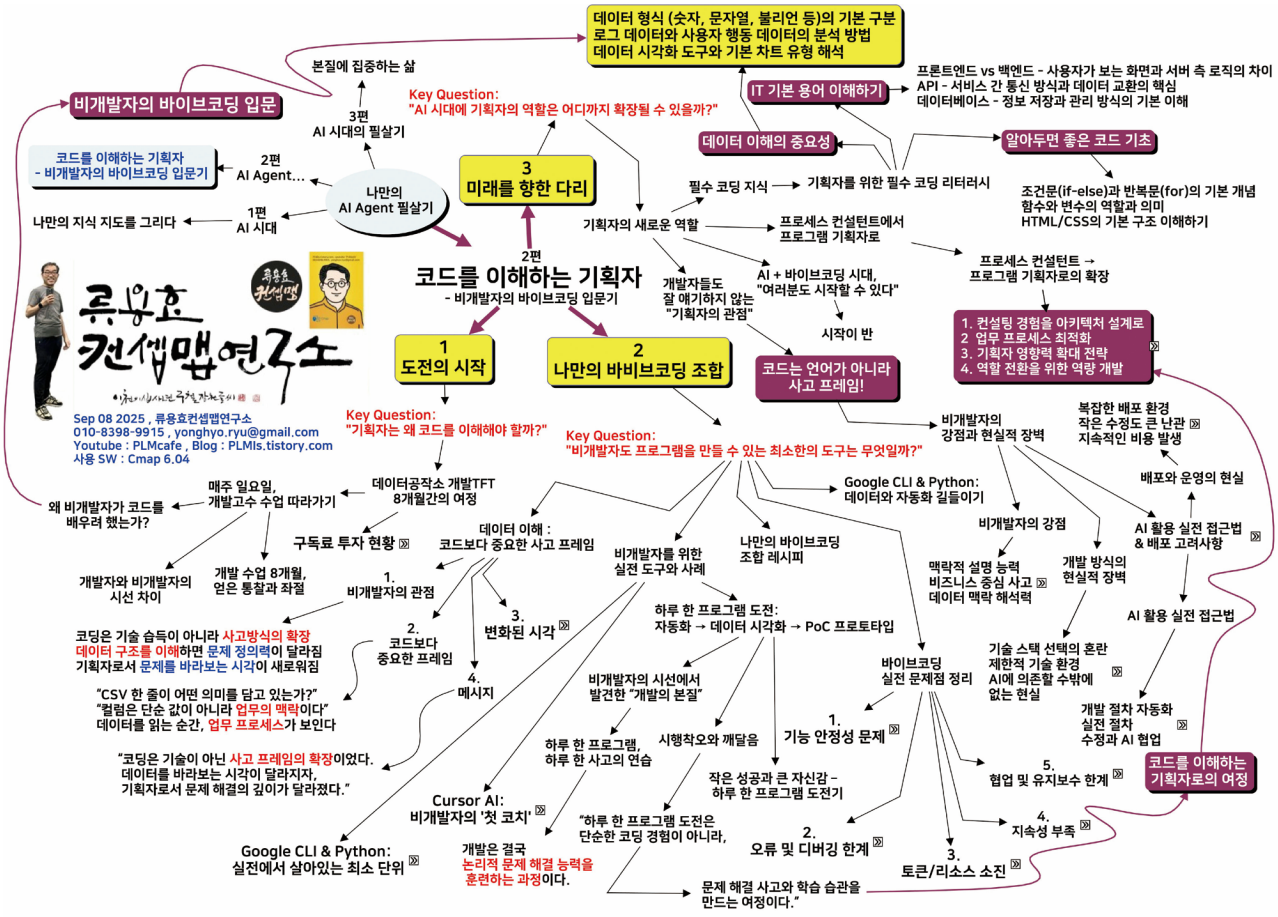
What I learned from the field No. 23

My own AI agent killer move Ⅱ - A planner who understands code, an introduction to vibe coding for non-developers

"Stand on the shoulders of giants and look at a wider world."
- Isaac Newton

[Context from previous section: . 23
My own AI agent killer move Ⅱ - A planner who understands code, an introduction to vibe coding for non-developers
"Stand on the shoulders of giants and look at a wider world."
- Isaac Newton

---]



October 2025

· 59

What I learned from the field No. 23

Why a non-developer wanted to learn code

The reason I, as a non-developer, started learning code was due to personal necessity. As a PLM construction PM, I wanted to communicate in the same language as developers, and while conducting process consulting, I wanted to verify system/process flows at the actual code level. I also wanted to implement ideas into prototype code by connecting concept maps and AI, and expand based on data and AI. A big motivation was also wanting to experiment with ideas quickly by easily creating prototypes through vibe coding.

There were various reasons why non-developers in general also wanted to learn code. It was to automate work by making repetitive and simple tasks efficient, to directly handle data structures to derive insights and strengthen data understanding. Coding was also necessary to understand technical language in the collaboration process with developers, to facilitate communication, to directly test and visualize ideas, and to

enhance creative problem-solving abilities. Additionally, it was to secure digital literacy and convergence capabilities to expand careers, and to adapt to the AI era through understanding code, which is a prerequisite for using AI tools.

In conclusion, the reason for non-developers to learn code was not to become developers, but to directly handle ideas, to experiment quickly, and to become better collaborators and creative problem solvers. The difference in perspective between developers and non-developers was clear. Developers focused on 'how to write code and logic' and were interested in performance, stability, and technical feasibility, while non-developers focused on 'why is this necessary' and considered usability, efficiency, and business value important. For example, when looking at the same CSV data, developers saw the structure and processing method of the data, while non-developers saw what the data was telling and how it could be used in management decisions.

"The future is not to predict, but to imagine."

- Alan Kay

"The wisest man is the one who continues to learn."

- Socrates

▲ A planner who understands code, an introduction to vibe coding for non-developers (Map by Ryu Yong Hyo)

[Context from previous section: "to imagine."

- Alan Kay

"The wisest man is the one who continues to learn."

- Socrates

▲ A planner who understands code, an introduction to vibe coding for non-developers (Map by Ryu Yong Hyo)

---]

60 ·
2025/10
COLUMN

My own vibe coding combination: Lessons from small successes

The new role of a planner in the age of AI and vibe coding has been significantly considered. Vibe coding is a concept first mentioned by Andrei Karpathy in February 2025, which was a method of programming by explaining the 'feel (vibe)' of the desired result to AI in natural language, rather than writing code. This reflected the trend of code writing skills transitioning to creativity and planning abilities. The AI development methodology for non-developers consisted of problem definition, PRD (Product Requirement Document) creation, AI prompting, and result verification. The planner focused on problem definition and user experience, clarified requirements by conversing with AI, refined the results, visualized ideas with quick prototypes, and focused on identifying improvements.

Over an eight-month journey, I developed my own AI application method, or 'killer move'. This was not simply a list of multiple paths, but organically connected on a single map. AI became not just a tool, but a collaborator in drawing this map. My AI killer move was as follows.

By combining these tools, I was able to implement data analysis automation scenarios and task automation bot construction scenarios. For example, I wrote data collection scripts with Cursor AI, cleaned and visualized data with Python, and scheduled regular executions with Google CLI.

Above all, understanding data was a more important thought frame than code. I realized that coding was not about acquiring skills, but expanding the way of thinking. Understanding the data structure changes the problem definition, and the perspective of looking at the problem as a planner became new. Understanding what a line of CSV means, and understanding that a column is not a simple value but the context of the task, I began to see the business process the moment I read the data. This changed perspective led me to question not just the results, but the flow and causes, enabled collaboration and design in the same language as developers, and led to quick experiments and verification based on data.

Through the 'One Program Challenge a Day', I accumulated small successes. I emphasized learning through experience and trial and error rather than perfection, and realized that the essence of development is practicing thinking. In other words, code is just a tool, the key is the ability to accurately understand and structure the problem, failure is learning, and small successes build the growth curve. I felt that the process of

constantly learning, recording, and reusing was much more valuable.

However, there were practical problems with vibe coding. There were problems with regression tests, where existing features were damaged when adding new features, and stability issues caused by AI not remembering the entire context sufficiently. There were also errors and debugging limitations due to infinite loops, incorrect logic creation, misunderstanding error messages, and excessive consumption of tokens/resources during the modification process. When sessions changed or contexts became long, AI forgot the detailed flow of the previous code, and there were limitations that the code written sporadically by AI lacked structure, making collaboration and maintenance difficult. Through experiencing these problems, I concluded that understanding code or collaborating with developers is essential.

Bridge to the future: The new role of the planner

In the AI era, the role of the planner could be greatly expanded. The strengths of non-developers were in data context interpretation, business-centered thinking, and contextual explanation abilities, and this clearly defined the meaning and relationship of CSV data columns.

■ Cursor AI: It played the role of the 'first coach' for non-developers. As a companion that lowers the wall of coding, it alleviated fears of complex grammar, errors, and environment settings. Cursor AI was not just about automatic code generation, but translated my intentions into code, enabling small experiments and iterations, and supported vibe coding learning. It was a GPT-4 based AI code editor compatible with Visual Studio Code (VS Code), allowing coding in natural language, providing immediate error correction, step-by-step explanations, and code refactoring features.

■ Google CLI (Google CLI): It was a new weapon for handling data and systems. It maximized speed and efficiency by automating repetitive tasks with commands instead of clicks. It controlled cloud resources such as virtual machines (VMs), storage, databases (DBs), handled data, and directly called APIs to facilitate service integration. It became a weapon for experts beyond the limitations of GUI.

■ Python (Python): It was the most useful minimum unit in practice. Its strengths were easy and intuitive syntax, vast libraries, and fast prototyping. Even small codes like reading/writing a line of data, simple automation scripts, etc., could have a big effect, and it was used for CSV analysis and visualization, task automation, AI·ML model experiments, etc. Cursor AI and Gemini were built-in for easier use.

"The secret of success is to be ready to seize the opportunity."

- Benjamin Disraeli

Page 4

October 2025
· 61
What I Learned on the Field No. 23
I explained the game, focused on business value and purpose rather than logic, and provided communication skills that explained the overall flow and context rather than technical details.
There was a need to expand capabilities from process consultant to program planner. I applied consulting experience to system architecture design, converted business analysis skills into system requirements, and designed a better UX (User Experience) through the integration of user perspective and system perspective. It was important to act as a bridge between the business department and the IT department, discover inefficient points through business process optimization, and solve system bottleneck phenomena from a data flow perspective. The ability to convert consulting deliverables into software specifications and verify optimization through workflow simulation was required.
The planner needed to strengthen negotiation skills with the development team based on technical understanding, build a data-based decision-making model, present an integrated perspective that connects business and technology, and secure the ability to concretize ideas into prototypes. The capabilities needed for this included system thinking, technical literacy (API, DB structure, basic concept of cloud service), agile methodology, and the ability to use collaboration tools such as Jira, Figma, Miro. It was important to break down the boundaries between planners and developers and build an integrated collaboration system to define and solve problems together.
The essence of AI was 'help', not 'subject'. AI executes without hesitation, but it was up to humans to judge whether it was the right direction. I realized the risk of security issues and human capacity degradation by entrusting tasks like meeting minutes summary to AI. Convenience does not always mean efficiency, and wrong dependence could lose important human abilities. So, I verified AI's answers at least three times, because setting the right direction was more important than quick execution. The answer given by AI was not the end, but the starting point.
My journey with AI made me constantly question myself. AI was not a machine to replace humans, but a companion to help humans enter a deeper world of thought and creation. The secret weapon I found was this. Thanks to AI, I was able to spend more time on my essence (core). Thanks to AI, which takes over simple repetitive tasks, I was able to focus on human-specific capabilities to think, plan, and judge. AI was no longer

an option, but an essential tool and collaborator.

What was important was how to connect this powerful tool with my essence, create my own unique value, and continuously execute deep thoughts about how to create the future. AI was a 'help' to those who had talent but hit a limit, opening the way to become an AI singer, AI film director, AI writer, AI programmer. Rather than just chasing efficiency, it was necessary to focus on the essence, read the flow of change, and continue to hone your own 'secret weapon'. The first step towards the future was to challenge right now.

Vibe coding resolved the gap between planning intentions and development execution, and discovered the expansion and possibilities of the planner's role in the AI era. It was possible to use time for creative work by getting out of repetitive work with task automation, and strengthen the ability to make data-based decisions and derive insights. It was important to start with making one program for 30 minutes a day, and courage to start was more important than perfection. However, it should not be forgotten that the pros and cons of vibe coding should be well understood and applied. Especially for simple programs for personal use, it's okay, but in the case of program development for external services, it must go through a code review by advanced developers to ensure there are no security issues, data leaks, etc. It should be remembered that AI is proficient in solving clearly defined problems, but cannot design a robust system by interpreting complex and ambiguous business requirements.

"I am not smart. I just wrestle with problems longer."

- Albert Einstein

"Coding is not a technology, but an extension of the thinking frame."
