SEQTEK

The Skill Shift: What Enterprise Developers Need to Learn Now

By Kenn Williamson · SEQTEK AI Practice

AI has removed writing code as a bottleneck — but the real differentiator isn't prompt engineering. It's problem framing, systems thinking, and communication. This guide breaks down exactly which skills enterprise developers need to thrive in the AI era.

INSIDE THIS GUIDE

What My Engineering Degree Actually Taught Me

I spent four years in mechanical engineering school learning how to write problem statements. Not how to solve problems. How to frame them.

At the time, it felt like busywork. My professors kept dragging us back to the same question: “What problem are you actually trying to solve?” That training has become more valuable in the last two years than anything I learned about thermodynamics or fluid mechanics.

When I work with developers who have traditional CS backgrounds, I notice a pattern. They often start from “here’s the problem we’re solving” as handed to them through requirements. They’re ready to implement. They want to get to the code. I keep going back to the customer: explain the actual problem, not the solution you want.

Classical computer science education often focuses on data structures, algorithms, and implementation details. Mechanical engineering taught me to think in systems — inputs, outputs, feedback loops, constraints. When so much implementation detail is being automated, systems thinking becomes the differentiator.

> ***“Many projects fail not because the team built the wrong thing badly — but because they built the wrong thing correctly.”***

The Doctor Metaphor

Working with stakeholders is like going to the doctor. A patient comes in saying their stomach hurts and they think it’s appendicitis. A good doctor doesn’t dismiss that research — but they also don’t skip the examination and go straight to surgery. They run tests. They ask questions. They investigate the actual problem based on symptoms.

A few months ago, I was working with a client in oil and gas. Smart technical people who know their domain cold. They came to me with a fully-formed solution. But when I started asking questions about the underlying problem, it became clear they hadn’t fully diagnosed it themselves. They’d jumped straight from pain to treatment.

The fix wasn’t what they originally proposed. It actually solved the problem they were experiencing — rather than the problem they thought they had. That’s what developers need to be doing now.

The Real Shift Isn’t About Syntax

The ability to write syntactically correct code from memory is becoming less valuable by the month. You still need to read and understand code. But memorizing method signatures and API patterns? That’s increasingly automated.

What’s not automated is the ability to look at a business problem and frame it correctly. To push back when a product owner hands you a solution disguised as a requirement. To recognize when the customer is describing a symptom rather than the actual disease.

A 200,000 token context window sounds huge until you actually work with a real application. When I started my personal website project, the model could hold the whole thing in context. Then I added Google OAuth, Amazon SES, a blog feature, complex relational database work. It didn’t take long before the model couldn’t see the whole application. People with massive legacy monoliths struggle with this. You have to build intermediate structures: indexes, abstractions, architectural documentation the model can reference to understand the larger scope.

> ***“The premium is shifting from ‘can you implement this algorithm from memory’ to ‘can you define the right problem, design the right system, and communicate effectively while you build it.’”***

The Communication Bottleneck

The JetBrains 2025 survey found that developers now rank internal collaboration and communication as important as technical tools for their performance. That matches what I’ve experienced.

I learned this lesson the hard way on a legacy rewrite project. Old VB.NET application, junior developers on the team, I was the architect. The project stalled. So I took the reins — in one week I basically redid the application, doing what would have taken two months five years ago. I thought I saved the day. It completely backfired.

We went from 60% finished to 99% finished in that one week. Then it took another three weeks to get to 100% because of all the fallout from my lack of communication. Three weeks to close a 1% gap. That’s where projects fail now — not in the implementation. In the communication.

Reading Code vs. Writing Code

I still spend most of my day reading code. What has shifted is what I'm looking for — not syntax errors. When I review AI-generated code, I'm asking four key questions:

Can I understand this?
If I can't understand the code, it needs to be rewritten. Generative models are non-deterministic — small prompt changes produce wildly different outputs. I need to be able to reason about what's happening.

Are we mixing patterns?
Across different context windows, the model gets pushed into different architectural patterns. If you let pattern mixing slide, you wake up with competing paradigms fighting each other in your codebase.

Was there a meeting of the minds?
Does the code actually do what it should? The AI has no context about users or the business. It can't know whether it understood your intent. That's your job.

Is this the right level of abstraction?
Recognizing what should be generalized and what shouldn't is something humans are still better at. The model may get the syntax right while missing the architecture entirely.

The Vibe Coding Problem

There's significant debate about 'vibe coding' — developers using AI to generate code they don't fully understand. The common issues: off-by-one errors, confusing logical branches, conditions so over-filtered that no case ever reaches them. How do you catch them? Testing. There's no substitute. Run tests, run the linter, run static analysis, then do manual testing and listen to your users.

The New Workflow

My day is round-robin. I typically have three or four different projects in flight simultaneously. Write a prompt, let the agent work, switch to the next project. Review. Set the next step. Switch again. I'm spending 70–90% of my time reviewing output and maybe 10–30% on actual prompting. The context switching is hard, but I'm not afraid to stop and investigate when something feels off.

What Actually Matters Now

Problem Framing

- Understanding what you're actually trying to solve before jumping to implementation. Pushing back on solutions disguised as requirements.

Communication

- Bringing people along. Explaining technical decisions in terms stakeholders understand. Not being the hero who saves the day and creates three weeks of fallout.

Architecture & Patterns

- Recognizing the right level of abstraction. Keeping patterns consistent. Understanding systems at a level above individual functions and files.

Reading & Reviewing Code

- Not for syntax — for clarity, consistency, and correctness at the architectural level.

Testing Rigorously

- Automated tests, manual testing, user feedback. The AI can't know if it understood your intent. Only testing reveals that.

> ***The skill shift is real — and it's happening faster than most developers realize. The winners won't be the ones who prompt best. They'll be the ones who frame the right problem, build the right system, and bring people along for the ride.***
>
> **— Kenn Williamson, Enterprise Architect, SEQTEK**

About the Author

Kenn Williamson

Enterprise Architect · SEQTEK

Kenn Williamson focuses on integrating AI solutions that empower organizations to optimize their workforce. By understanding business needs through thorough discovery processes, he designs complex systems that effectively address those needs. His role also involves mentoring development teams to deliver innovative solutions that drive success.

About SEQTEK

SEQTEK is your local partner for business consulting services — helping organizations innovate, implement, and deliver a better tomorrow with expert Tulsa business consulting. Purpose-driven and personal, SEQTEK brings a touch that other consulting companies don't bring to the table, and they always deliver on what they say they will do. Through our data and strategy practice, we align people, processes, and technology to solve complex challenges and unlock sustainable growth. Our consultants combine strategic insight with hands-on execution to help businesses move forward with confidence.

Ready to talk AI strategy for your organization?

Let's find the right AI opportunities — together.

Talk to SEQTEK → seqtek.com