SOFTWARE CENTAURS: BECOME AN AI-AUGMENTED ENGINEERING TEAM WITHOUT SACRIFICING DISCIPLINE

CHIRON CODEX

LLM-based coding assistants—or the companies that sell them—promise huge advances in developer productivity. But without engineering rigor, this translates to accelerating the rate at which you accumulate technical debt, open bugs, and dissatisfied customers. Learn how to incorporate AI tools into your team's processes while staying in control of the engineering lifecycle, instead of outsourcing your team's capabilities to a computer.

"You're absolutely right, that's on me"—or is it?

Another day, another report of a coding agent pushing changes to the production database, deleting backup data, or pushing broken software to main. It's enough to make you think that maybe these AI tools aren't all they're cracked up to be. At the same time, the data show that organisations that adopt AI increase their delivery throughput.[1] How can both be true? The fact is that AI acts as an amplifier, improving your software team's strengths—and simultaneously exposing and worsening their weaknesses. The key to AI-augmented software engineering is to focus on the foundations; to make sure that your team know how to successfully deliver quality software, so that the AI tools can help them to supercharge that success.

[1] https://dora.dev/research/2025/dora-report/

The Problem: who knows where the value goes?

You've got your minimal viable product out, and the market's reacting positively. So are investors; you've closed your Series A and the board has given you clear goals to grow the business. Your team needs to completely change focus now: the quick prototypes that rapidly validated your proposition are now technical debt, and you need to concentrate on scaling your solution while still iteratively adding business value.

AI augmentation through LLM-based coding assistants, or even autonomous agents, sounds like a dream; each developer on your team can orchestrate a team of agents and get an order of magnitude more work done. You'll fly through your feature backlog, replatform onto a scalable infrastructure, and clear up all that technical debt, maybe without even growing headcount!

Unfortunately, when you try it, the reality doesn't live up to the hype. The agents generate new features at a phenomenal rate, but the code doesn't integrate cleanly, the team can't keep up with testing, and deployments fail. You ask your lead engineer how to get the build back to green and they don't know; they paste your question into their coding assistant and see what it comes up with.

Your team has amplified its capabilities, sure, but now it's just a much faster seed-stage startup. You still need to realign the team on value streams, backed by engineering rigor—you need your tools and processes to bring your team forward in the right direction, not to do whatever they were doing, but faster.

The Solution: software centaurs

Patterns in AI-Augmented Software Development is, at its core, a feedback loop for AI-augmented software engineers. It avoids the vibe-coding trap by giving engineers reusable approaches to reflect on and and improve the way that they work with their coding assistants—including *You Reflect*, a pattern that helps them take a step back and consider the opportunities for getting more out of the tools.

Did the language model report that it had completed a task with "100% production-ready code"? That doesn't matter, *You Review* the code anyway, or maybe the *Model Reviews* it if you've got to a point where that works. Either way, *Trust the Tests*, not the language model, to tell you when the software works.

Does your team understand a story well enough to start coding? They could *Just Ask* to probe their ideas and answer questions, *Ask for Alternatives* to evaluate different ways to implement the feature, or *Disclose Ambiguity* to discover the gaps in their knowledge, and the questions they need to answer—or get the customers to provide answers to.

The patterns approach, combined with a solid understanding of engineering fundamentals and an understanding of how people work with software based on deep research in the field, gives your team the foundation they need to grow their capabilities successfully—and then to supercharge them into high-performing software centaurs.

The Platform: a roadmap to a centaur engineering culture

I meet your team, identify the pain points in their current process, and the concerns they have in adopting coding assistants and agentic software engineering tools. This information feeds into a bespoke roadmap your team follows to unlock AI-augmented productivity: where to start; how to adapt their processes; what feedback to measure.

Working with your team, we establish a solid, reliable engineering foundation that they can accelerate by judicious adoption of AI tools. "Velocity" is a vector, both an amount and a distance; we make sure your team's pointing in the right direction as we hit the accelerator.

The Business Case: Amplify your advantage, address your gaps

The Chiron Codex process isn't just about adopting the latest tools—we re-engineer your team for efficient, reliable delivery of valuable software. Technical debt is a "vicious cycle" that sucks up productivity, and reduces opportunity for investment in new initiatives.[2] Rework—fixing bugs, redesigning systems, starting over—costs the average engineering team 20%-40% of their time, while a top-quartile team can get that down to 10%.[3]

Giving a team AI tools without establishing a high-performance delivery capability lets them ramp up the speed at which they write new software, without giving them everything they need to write valuable software. Stop watching your token spend, and instead measure reliability and value delivery.

Your Next Step: The Engineering Discipline and AI Workflow Audit

Visit https://cal.com/chironcodex/office-hours to book a no-obligation discovery call, where you can arrange your first workflow audit.

In that audit I meet your team for a day, facilitating a combination of one-on-one and whole-group blameless feedback sessions in which I understand how the team works, their pain points when it comes to delivering reliable, valuable software, and their attitudes towards AI adoption. The outcome of these sessions is an understanding of the strengths and weaknesses of the way that the team works, and the opportunities and risks associated with augmenting their work using AI-based tools. You receive a bespoke report with a risk-focused prioritization of your team's processes and capabilities, the steps to take to point the team in the correct direction while boosting their performance, and the measurements to record to track progress.

You can take that report and work with your team to improve their outcomes, or engage me on a retainer basis to mentor your engineers, guide changes to their processes, and adapt the roadmap as their capability improves, and new opportunities and problems arise.

---

[2] https://www.mckinsey.com/capabilities/tech-and-ai/our-insights/breaking-technical-debts-vicious-cycle-to-modernize-your-business

[3] https://reworkcost.com

About the Author

Dr. Graham Lee is an accomplished software engineering author, with more than 20 years' experience improving software and documentation at Apple, Facebook, ARM, growth-stage startups, and beyond. He's the author of multiple technical titles on software engineering, including *Professional Cocoa Application Security*, *Test-Driven iOS Development*, *The Python Workshop*, and *Modern Programming: Object-Oriented Programming and Best Practices*, with *Patterns in AI-Augmented Software Development* coming soon.

He gained his D.Phil. in Computer Science from Oxford University in January 2026, where he studied the role of Research Software Engineers in improving academic software outcomes. Currently, Graham runs Chiron Codex, an initiative to help software engineers become centaurs through thoughtful application of AI-augmented tools. He also takes a deep dive through the history of software engineering, sharing his discoveries with the community on the Structure and Interpretation of Computer Programmers podcast: https://sicpers.info/podcast.

CHIRON CODEX