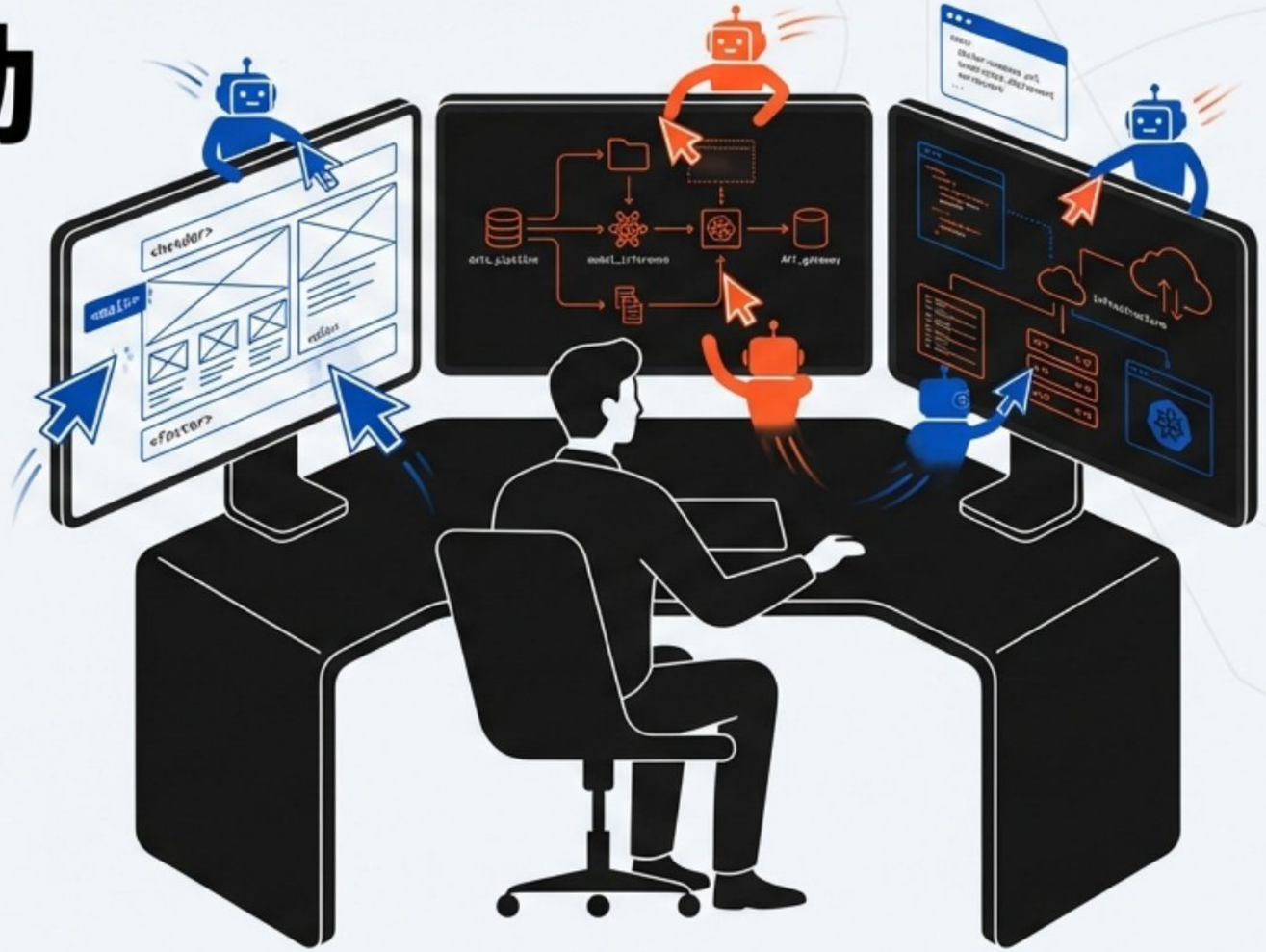
把 AI 當成資深同事使喚：我的自動化開發實戰指南



- 從「親自寫 Code」到「指揮 AI」的真實痛點與解法。
- 打開引擎蓋，看看現代 AI 工程學的工作流底層。

NotebookLM

這不是一場教你怎麼寫 Prompt 的新手課



如何寫出完美的 Prompt 咒語

比較哪一家的 LLM 模型比較聰明

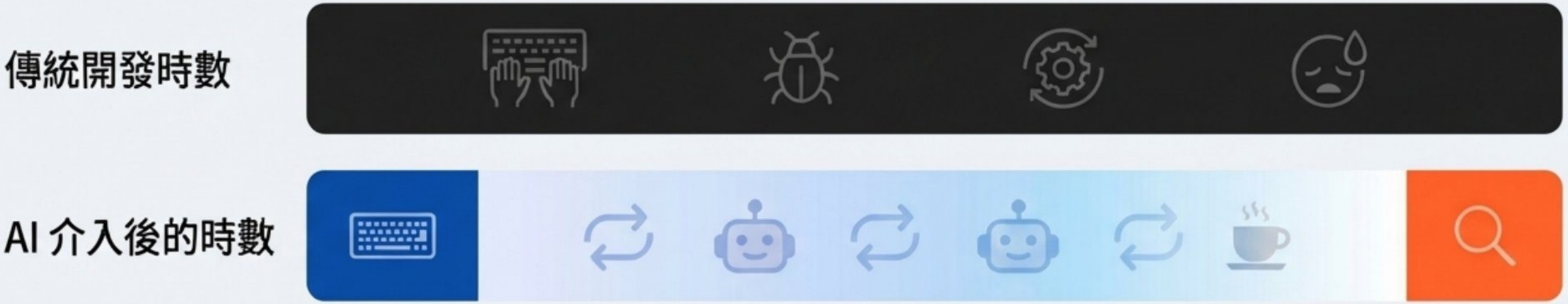


如何把 AI 當作不知疲倦的初階工程師來協作

建立一套能讓 AI 穩定輸出、減少 Review 負擔的工作流

NotebookLM

讓 AI 在背景處理繁瑣開發，我只負責驗收成果



- 丟出需求後，讓 AI 長時間自動撰寫程式碼。
- AI 自行執行、測試、反覆修 Bug，直到通過測試基準線。

NotebookLM

AI 同時也是拯救行政庶務的超強秘書



- 一句話觸發完整分析：「幫我整理 APP Team 本週工作狀況」。
- AI 自動交叉比對 GitLab Issues 與 Slack 歷史訊息。
- 主動識別潛在風險：自動找出逾期未解項目與被忽略的關鍵討論。
- 釋放管理與溝通成本，將團隊精力重新聚焦於核心商業價值創造。

NotebookLM

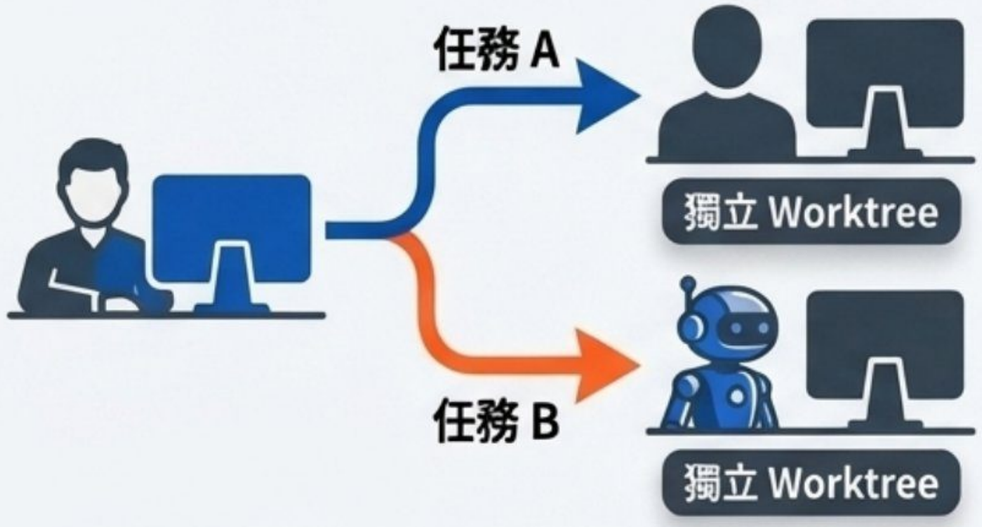
傳統 Git vs. Git Worktree 工作流程差異

傳統 Git 工作流程（序列式）



- 單一工作目錄，任務需序列執行
- 頻繁 stash/pop，易混淆或遺漏
- 切換分支耗時，且會變動當前目錄

Git Worktree 工作流程（平行式）



- 多個獨立工作目錄 (Wrls)，任務平行進行
- 無需 stash/pop，每個目錄狀態獨立
- 直接在不同目錄工作，無需切換分支

NotebookLM

MCP 給了 AI 看見外部世界與動手的權限



- 為什麼 AI 能讀 GitLab、發 Slack？因為 MCP（Model Context Protocol）。
- 標準化的外部工具整合協議，取代過去痛苦的 Custom API 串接。
- 從被動的「文字生成器」變成主動的「系統操作者」。

NotebookLM

常用的 Agent Skills 範例

💻 代碼風格與品質 (Code Style & Quality)

- Linters (e.g., ESLint, Pylint)
- Formatters (e.g., Prettier, Black)

🚀 測試與部署 (Testing & Deployment)

- Unit Testing (e.g., Jest, pytest)
- CI/CD Pipelines (e.g., GitHub Actions, GitLab CI)

📘 文檔與知識庫 (Documentation & Knowledge Base)

- API Documentation (e.g., Swagger/OpenAPI)
- Internal Wikis (e.g., Confluence, Notion)

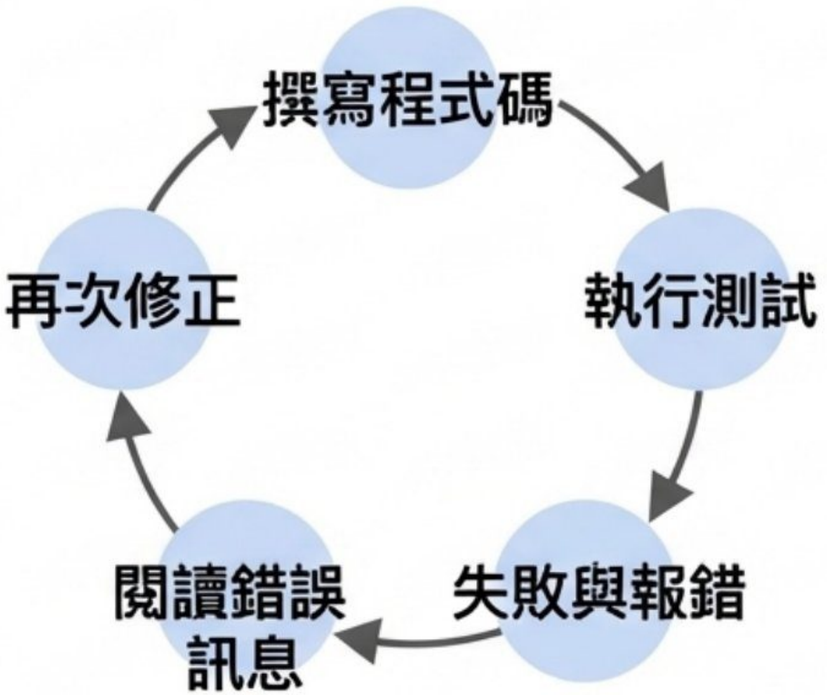
🧰 特定領域工具 (Domain-Specific Tools)

- Database Schema Management (e.g., Prisma, SQLAlchemy)
- Cloud Infrastructure (e.g., Terraform, AWS CDK)

NotebookLM

Ralph Loop 讓 AI 學會不屈不撓的自我除錯

- 撰寫程式碼 讓 AI 學會不屈不撓的自我除錯
- 執行測試 →請仔細分析失敗與報錯
- 閱讀錯誤訊息 並重複有測試修正



Ralph Loop 提示詞範例：

你現在是一位資深後端工程師。請執行專案目錄下的測試指令 `[測試指令]`。若測試失敗，請仔細分析終端機輸出的錯誤訊息 `[錯誤訊息]`，並自動修改 `[專案路徑]` 下的相關程式碼。

請重複執行這個流程：修改程式碼、執行測試、 分析錯誤，直到所有測試都通過為止。 最後請回報修正的內容。

NotebookLM

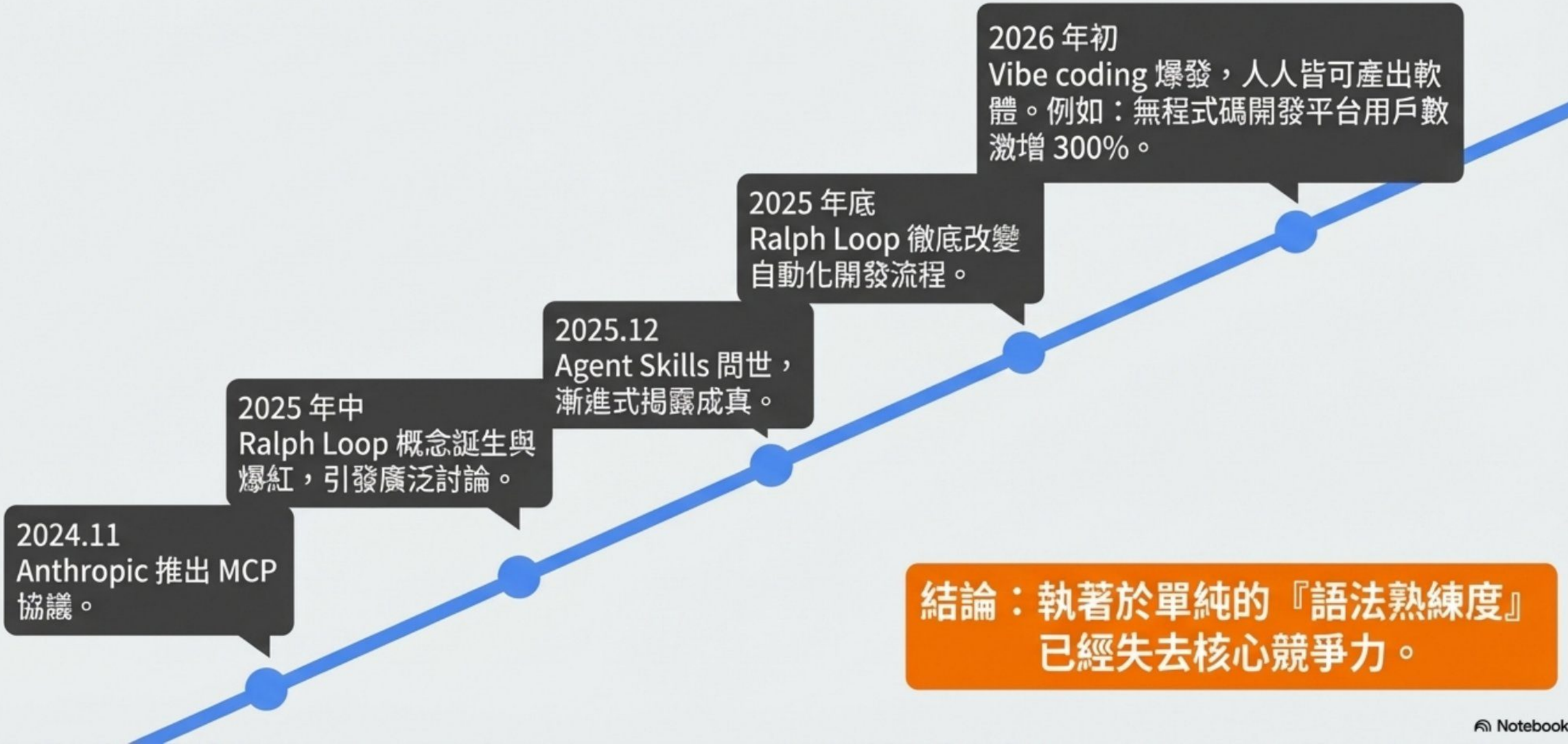
Superpowers 框架把軟體工程紀律強勢注入 AI 工作流



- 拒絕一次性生成：將 TDD（測試驅動開發）等嚴謹紀律整合進 AI 工作流。
- 確保 AI 先寫出會失敗的測試（Red），再寫最少程式碼通過（Green），最後重構（Refactor）。
- 讓 AI 產出的不僅是「會動的 Code」，而是「有紀律且可維護的架構」。

NotebookLM

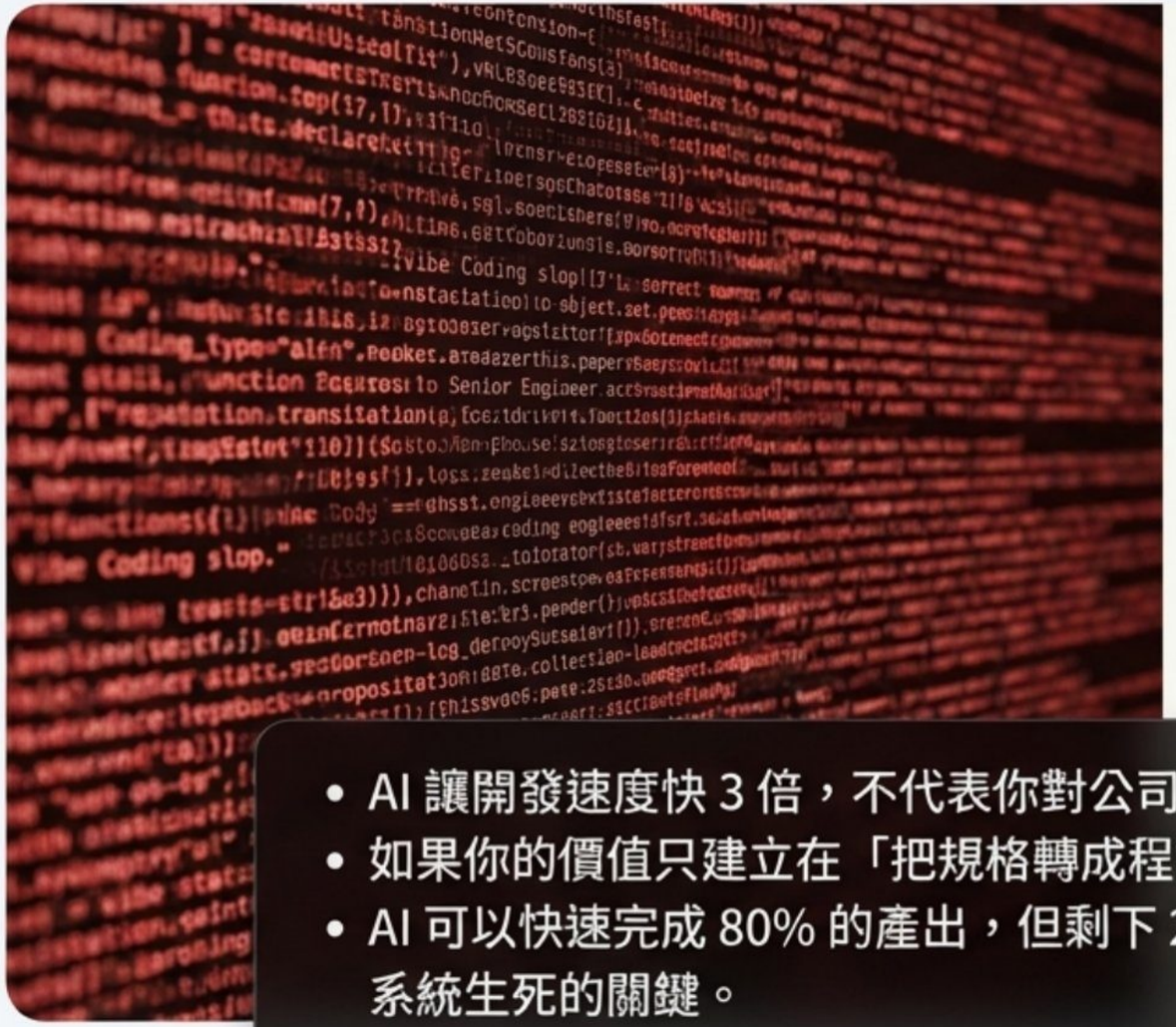
技術迭代速度，已經超越了人類學習語法的速度



結論：執著於單純的『語法熟練度』已經失去核心競爭力。

NotebookLM

殘酷的真相：未來最不缺的就是能寫程式的「執行者」



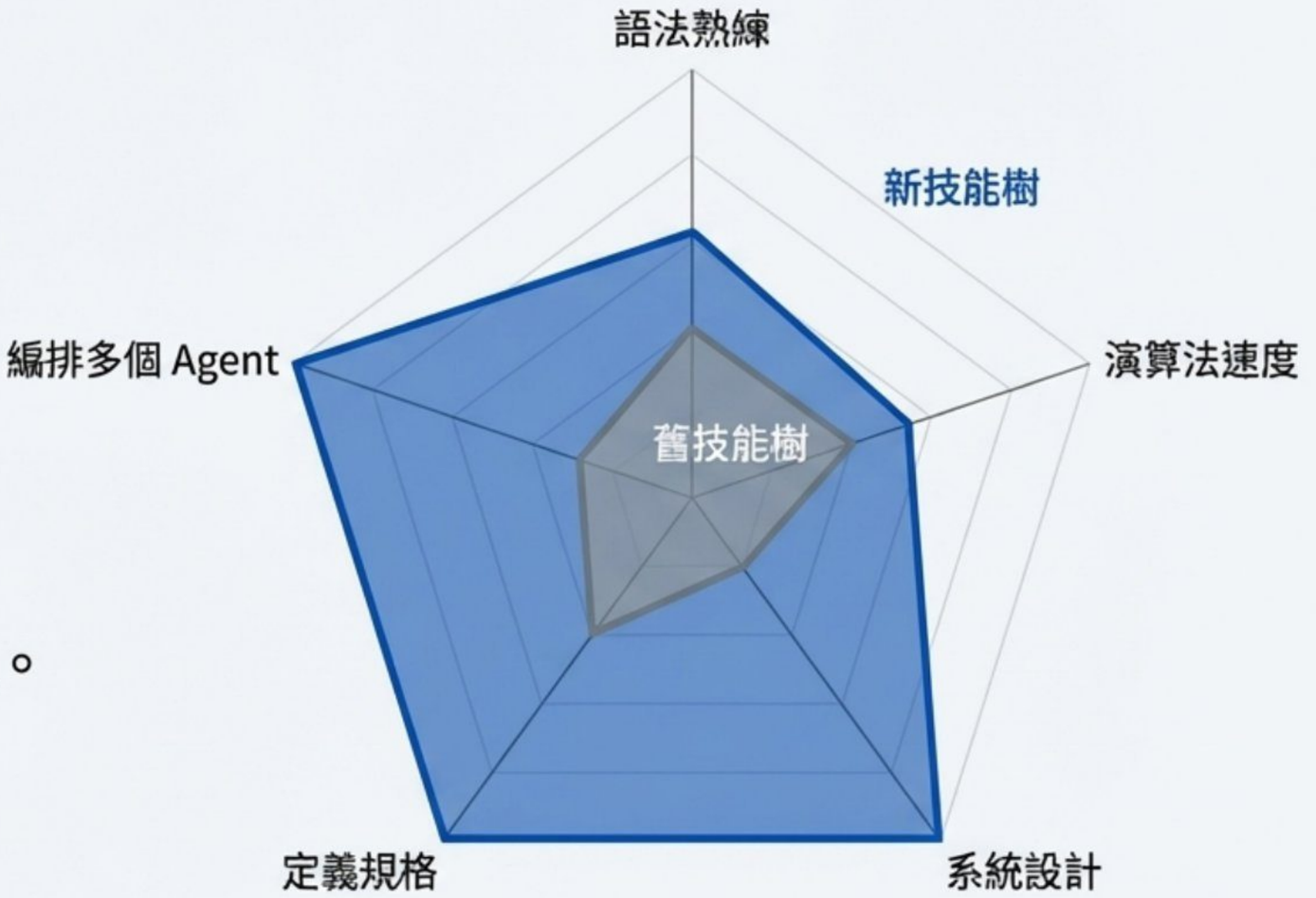


- AI 讓開發速度快 3 倍，不代表你對公司的價值提升 3 倍。
- 如果你的價值只建立在「把規格轉成程式碼」，那你非常危險。
- AI 可以快速完成 80% 的產出，但剩下 20% 的品質把關與系統設計，才是決定系統生死的關鍵。

NotebookLM

最搶手的人才畫像：從「工匠」進化為「總指揮」

- 舊思維：自己動手刻每一個細節。
- 新價值：定義系統架構、設定 AI 的目標與護欄、驗證產出品質。
- 工程師不再只是「寫程式碼的人」，而是「建構能寫程式碼的系統」的人。



NotebookLM

懂『為什麼要做』比懂『怎麼做』更具毀滅性的優勢

- 商業思維：AI 可以做「什麼」，但「該不該做」只有人能判斷。
- 懂得收集數據、擬定目標策略（OKR）、活用敏捷方法。
- 將省下來的時間，拿去理解客戶痛點、商業模型與毛利率影響，做出更好的技術決策。



NotebookLM

溝通能力，是驅動 AI 艦隊的唯一方向盤



跟 AI 溝通

跟人類溝通

將模糊的商業需求，精準拆解成 AI 聽得懂的 **Specs** 與結構化文件（Spec-Driven Development）。

AI 不能幫你在週會上說服老闆改變優先先級，也不能處理「客戶說的跟想的不一樣」的真實世界難題。

AI 取代的是「動手」，取代不了「動腦」和「動口」。

NotebookLM

~~「AI 可以幫我做什麼？」~~

「我要怎麼用 AI 來達成我的目標？」

拋棄被動思維，主動定義你的戰場。

AI 是強大的協作者，而不是替代者。善用它，讓我們專注於更有創造力與價值的挑戰。

NotebookLM