Code Quality and Security in an AI-Powered SDLC
AI 驱动的软件开发生命周期中的代码质量与安全

Using static analysis to keep the speed without losing control
用静态分析在提速的同时不失掌控

Code Quality and Security in an AI-Powered SDLC
AI 驱动的软件开发生命周期中的代码质量与安全

Using static analysis to keep the speed without losing control
用静态分析在提速的同时不失掌控

What do we actually mean by code quality?
我们所说的代码质量到底指什么?

Dimension ｜ 维度	What it means in practice ｜ 实际含义是什么
Correctness ｜ 正确性	Does it do the right thing, including edge cases 是否做对了事情，包括边界情况
Maintainability ｜ 可维护性	Can the next person change it safely and quickly 下一位接手的人能否安全、快速地修改它
Security ｜ 安全性	No exploitable weaknesses or unsafe dependencies 没有可被利用的漏洞或不安全的依赖
Test coverage ｜ 测试覆盖率	Behaviour is pinned down, regressions get caught 行为被固化，回归问题能被发现
Consistency ｜ 一致性	One shared standard, not per-author preference 统一标准，而非各写各的

Quality is a business number, not a style debate
质量是一个业务数字，而非风格之争

$2.41T

annual cost of poor software quality in the US
美国每年因软件质量低下造成的成本

$1.52T

of that is accumulated technical debt
其中属于累积的技术债务

$9.44M

average cost of a single US data breach
美国单次数据泄露的平均成本

Defects leave engineering and land on the business as outages, breaches, missed dates and slower delivery. Quality is a cost-control lever.
缺陷一旦流出研发环节，就会以宕机、数据泄露、延期交付和交付变慢的形式转嫁给业务。
质量是控制成本的杠杆。

CISQ: The Cost of Poor Software Quality in the US, 2022 report
CISQ：《美国软件质量低下的成本》2022 年报告

AI is now the default way code gets written
AI 已成为编写代码的默认方式

90%

of tech professionals use AI at work
的技术从业者在工作中使用 AI

80%

of new GitHub users tried Copilot in week one
的 GitHub 新用户在第一周就尝试了 Copilot

1M+

pull requests authored by Copilot coding agent, May to Sept 2025
由 Copilot 编码智能体提交的 PR 数量（2025 年 5 月至 9 月）

Adoption is settled. The open question is what happens downstream of the keyboard, where review and release still have to absorb the volume.
AI 采用已成定局。
真正的问题在于代码写完之后：评审和发布环节仍要消化这些新增的量。

Google Cloud 2025 DORA Report; GitHub Octoverse 2025
Google Cloud 2025 DORA 报告；GitHub Octoverse 2025

What AI changes about code quality
AI 改变了代码质量的哪些方面

- AI code is not worse everywhere, it fails differently
  AI 生成的代码并不一定更差，但是出错的方式各不相同
- 1.75x more logic and correctness errors
  逻辑与正确性错误多 1.75 倍
- 1.64x more code quality and maintainability issues
  代码质量与可维护性问题多 1.64 倍
- 1.57x more security findings
  安全问题多 1.57 倍

```
// The question is not
// 问题不是
// “Was this written by AI?”
// “这是不是 AI 写的？”

// The question is
// 问题是
// “Does this meet our
// “这是否符合我们的
//  quality standard?”
//  质量标准？”
```

CI answer ｜ CI 给出的答案

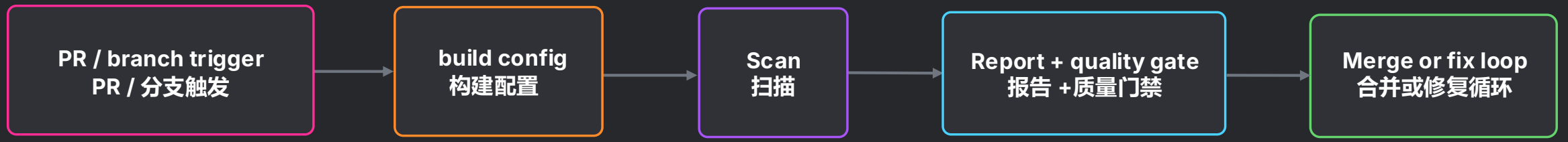
Apply the same deterministic policy to every change — whether authored by a human, an AI assistant, or both.
对每一次改动都应用同一套确定性策略——无论作者是人类、AI 助手，还是两者协作。

Source: CodeRabbit, 470 real-world pull requests, Dec 2025
数据来源：CodeRabbit，470 个真实 PR 样本，2025 年 12 月

Turning AI risk into standardized policy
把 AI 带来的风险转化为标准化策略

AI risk ｜ AI 风险	Static Analysis control ｜ 静态分析的应对措施
Plausible but flawed code ｜ 看似合理却有缺陷的代码	Inspections + severity policy ｜ 代码检查 + 严重级别策略
Copied local patterns ｜ 照搬局部代码模式	Duplicate code + code smells ｜ 重复代码 + 代码异味
Untested changes ｜ 未经测试的改动	Fresh-code coverage thresholds ｜ 新增代码覆盖率门槛
Risky dependencies ｜ 有风险的依赖	Vulnerability + license checks ｜ 漏洞 + 许可证检查
Legacy noise ｜ 历史遗留噪音	Baseline existing issues ｜ 为存量问题设基线

Where static analysis runs in the pipeline
静态分析在流水线中的位置



Pipeline's role ｜ 流水线的角色

Run the pipeline, show results, enforce build outcomes, and give developers a repeatable feedback loop.
运行流水线、展示结果、强制执行构建结论，为开发者提供可复现的反馈闭环。

Static Analysis' role ｜ 静态分析的角色

Analyze code and convert quality standards into standardized policy.
分析代码，把质量标准转化为标准化策略。

AI and static analysis are complementary
AI 与静态分析是互补关系

Static analysis is good at ｜ 静态分析擅长

Deterministic rules, whole-repository coverage, security and license checks, and the same verdict on every run. Cheap, fast, auditable.
确定性规则、全仓库覆盖、安全与许可证检查，每次运行结论一致。
成本低、速度快、可审计。

AI review is good at ｜ AI 审查擅长

Intent, cross-file reasoning, unusual patterns and suggesting fixes. Strong on context, weaker on repeatability.
理解意图、跨文件推理、发现异常模式并给出修复建议。
上下文理解能力强，但可复现性较弱。

Takeaway ｜ 要点 Use static analysis as the deterministic floor, and AI for judgement on top. Neither replaces the other.
把静态分析作为确定性的底线，在此之上用 AI 做判断。两者互不替代。

Static analysis makes AI review cheaper
静态分析让 AI 审查更省成本

First 首先

deterministic scan catches what rules can decide
确定性扫描先解决规则能判定的问题

Then 然后

AI reviews only what is genuinely left over
AI 只评审真正剩下的部分

Result 结果

fewer tokens, faster feedback, same coverage
更少的 token 消耗、更快的反馈、同等的覆盖率

Why it works
为什么有效

Research combining lightweight static analysis with LLMs cut the required LLM calls by 72 to 97 percent.
研究表明，将轻量级静态分析与大语言模型结合，可减少 72% 至 97% 的 LLM 调用量。

Rules are free to re-run. Inference is not. Spend tokens only where rules cannot decide.
规则可以免费重复运行，推理不行。把 token 花在规则无法判定的地方。

Operating model for AI quality in CI
CI 中 AI 质量保障的运作模型

Automated Tests
自动化测试

Static Analysis and code quality gates
静态分析与代码质量门禁

Security scanning
安全扫描

Policy-as-code
策略即代码

Repeatable CI pipelines
可复现的 CI 流水线

Shared Foundation
共享基础设施

Deep traceable history
可追溯的完整历史