Compound Engineering

AI時代のソフトウェア開発における「複利」の魔法

Presented by Every Inc. | The Future of Coding

従来の開発：技術的負債の蓄積

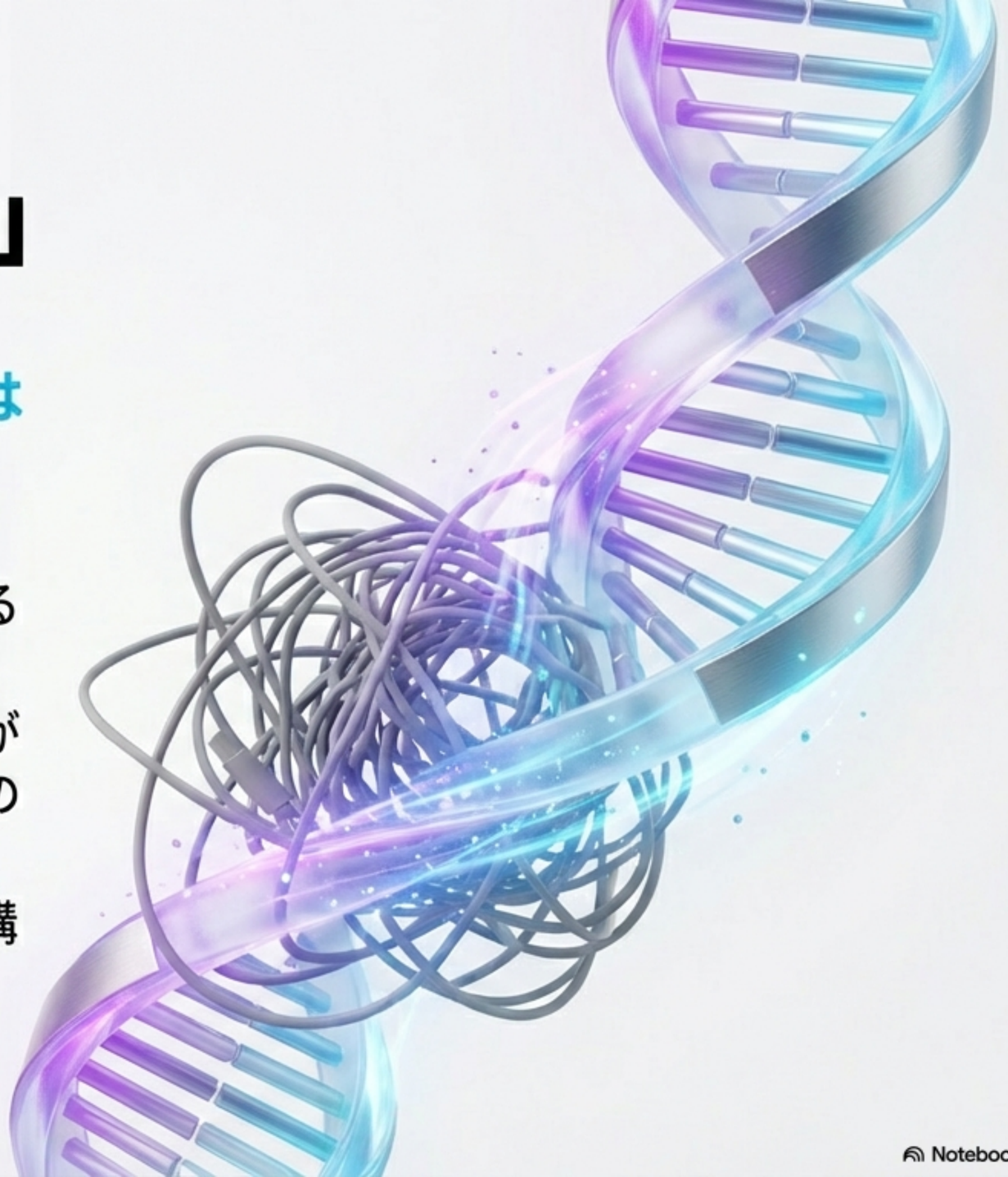
「機能Aを追加すると、機能Bの実装は『より難しく』なる」

- **エントロピーの増大**：コード量が増えるほど、エッジケース、依存関係、複雑性が指数関数的に増加する。
- **速度の低下**：開発チームは機能追加よりも、既既存コードの解読や修正に時間を奪われる。
- **結果**：開発ベロシティは時間とともに必然的に低下する。

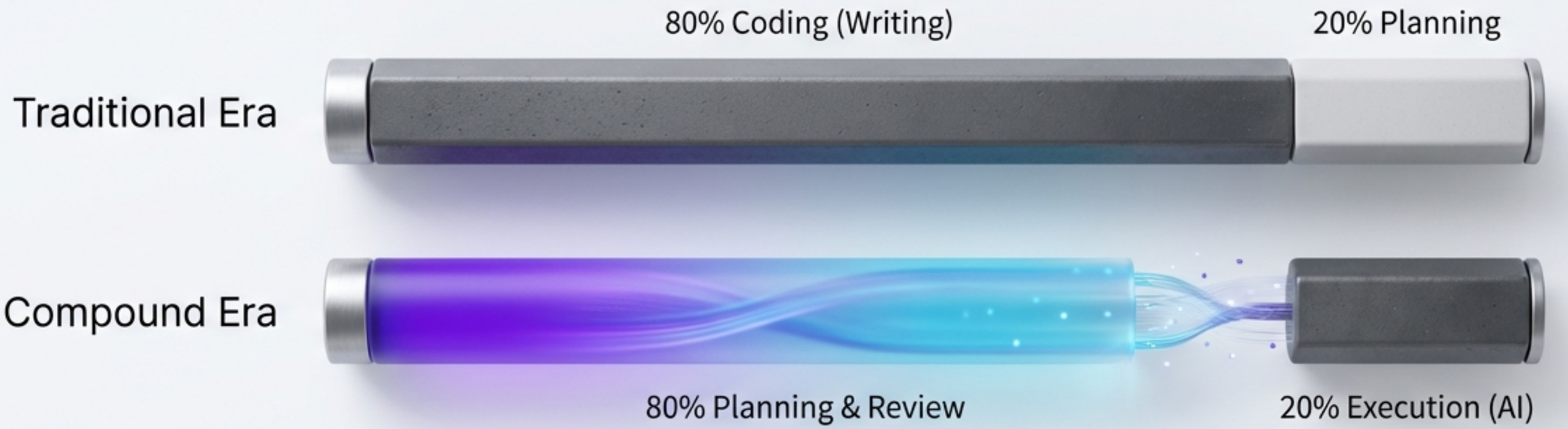
技術的負債の「逆転」

「機能Aを追加すると、機能Bの実装は『より簡単に』なる」

- **資産としての複雑性**：コードベースが成長するほど、AIの知識（コンテキスト）も成長する。
- **学習ループ**：バグ修正や設計判断をシステムが記憶し、次回の開発時に、エージェントがその知識を自動的に活用する。
- **定義**：AIエージェントのための学習ループを構築する新しいエンジニアリング手法。



開発プロセスの再定義：80/20の法則



手動でのコーディングはもはやボトルネックではない。

人間が注力すべきは「何を作るか（Plan）」と「品質の監査（Review）」。

書く時間を減らし、設計と指導に時間を割くことで、品質と速度が同時に向上する。

4つのステップ：循環する知能

1. Plan
リサーチと設計

2. Work
自律的な実装

3. Review
多角的視点による監査

4. Compound
知識の結晶化

単なる直線的なプロセスではなく、ループするごとにシステムが賢くなるサイクル。
AIエージェントが実行し、人間が監督・指導する。

Plan：すべては計画から始まる

/workflows:plan

- **リサーチ**：エージェントはコードを1行も書く前に、コードベースの歴史とインターネット上のベストプラクティスを調査する。
- **共有メンタルモデル**：実装内容、アーキテクチャ、成功基準を詳細なスペック（GitHub Issue/File）に落とし込む。
- **人間の役割**：AIに「何をすべきか」を正確に伝えるための高度な思考と指示。

Work：自律的な実装と反復

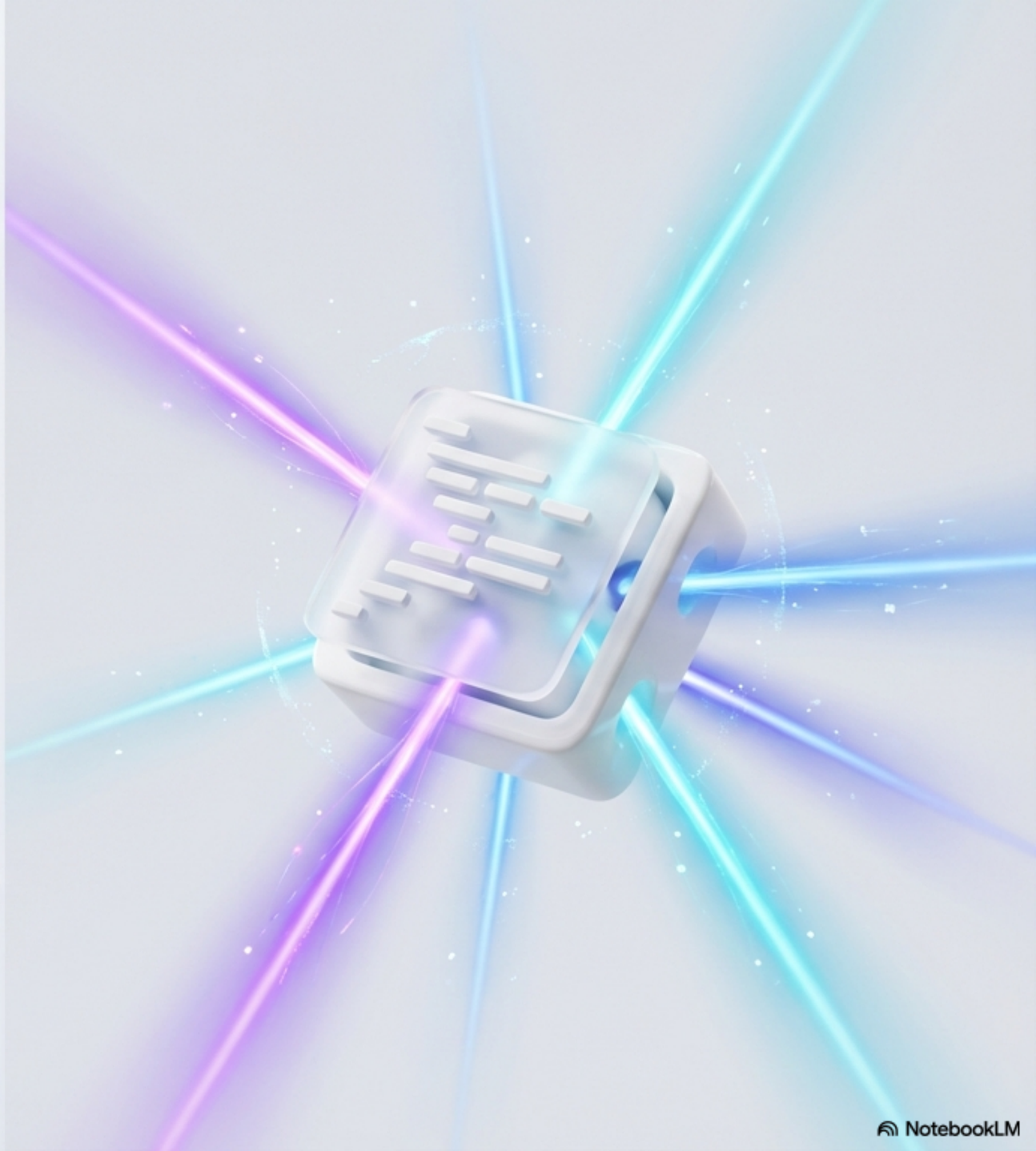
/workflows:work

- **実行**：エージェントが計画をToDoリストに変換し、ステップバイステップで実装。
- **視覚的な認識**：PlaywrightやXcodeBuildMCPを使用し、ユーザーのようにアプリを操作・視認しながら開発する。
- **自律改善**：プロトタイプを作成し、計画と一致するまで修正を繰り返す.

Review：エージェントスウォームによる監査

/workflows:review

- **並列処理**：単一のAIではなく、12以上の専門サブエージェントが同時にコードをチェック。
- **多角的視点**：
  - セキュリティ脆弱性の検出
  - パフォーマンスのボトルネック特定
  - 過剰エンジニアリング（Bloat）の防止
- **人間の役割**：AIが提示した指摘を統合し、最終的な採用判断を下す。



Ethereal Tech

Compound：知識の結晶化

/workflows:compound

- 核心的な問い：「同様の問題を以前に解決したことがあるか？」
- 資産化：バグの原因、設計の意図、プロジェクト特有のルールをCLAUDE.md 等のプロンプトファイルに記録。
- 効果：新しいエージェント（および新入社員）は、チームの集合知を即座に利用可能になる。失敗は二度繰り返されない。

The Compound Engineering Plugin

Official Claude Code Plugin / Open Source (MIT)

- 24の専門エージェント：役割分担されたAIのチーム
- 13のスラッシュコマンド：ワークフローを制御するインターフェース
- 自動化スキル：Playwright , Context7などのMCPサーバー統合
- 互換性：Claude Codeに加え、OpenCode / Codexへの変換もサポート

実証された効果：1人で5人分の生産性

Case Study: Every Inc. の事例

- **5つのプロダクト**：それぞれを1人の開発者が運用
- **数千人のユーザー**：デモではなく、実稼働している商用サービス
- **ゼロ・ボイラープレート**：手動でのテスト記述や定型コード作成はゼロ
- **結論**：適切なシステムがあれば、1人の開発者が数年前の5人分の成果を出せる。

導入：今すぐ複利を始める

```
> /plugin install compound-engineering
```

1. Install

Claude Code環境にプラグインを追加。

2. Sync

スキルや設定をOpenCode/Codexと同期可能。

3. Start

まずは /workflows:plan から始め、AIとの「計画」の威力を体験する。

エンジニアの進化:「書く」から「指揮する」へ

- 点滅するカーソルを見つめる時間は終わった。
- エンジニアは「知能のアーキテクト（Architects of Intelligence）」になる。
- 価値の源泉は、構文の暗記から、システム設計能力とAIへの「教育」能力へとシフトする.

Start Compounding Today.

コードを書くたびに、未来の開発が楽になる。

GitHub: EveryInc/compound-engineering-plugin