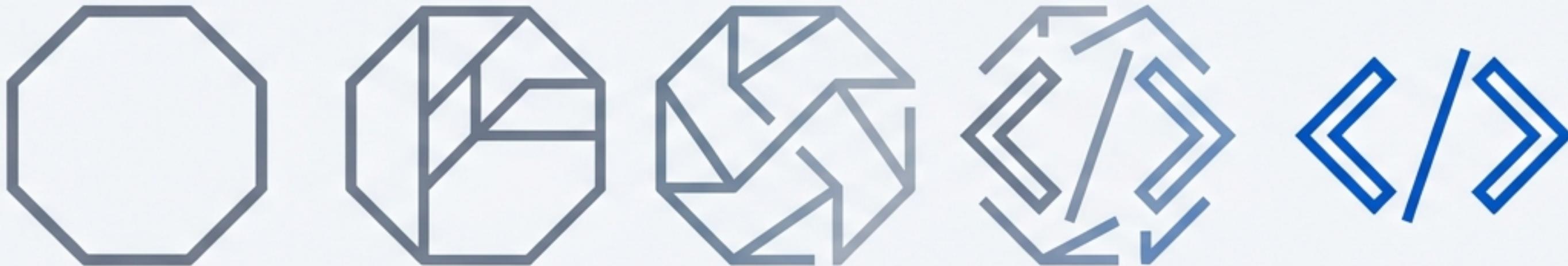


AI Factory v2.7



GitHub Issueをコードに変換する、自己改善型・自律開発パイプライン

現代の開発現場が抱える「摩擦」



繰り返し作業

テスト作成、実装、レビュー、PR作成などの定型業務。



品質管理の負担

手動レビューとテスト実行に時間がかかる。



知識の散逸

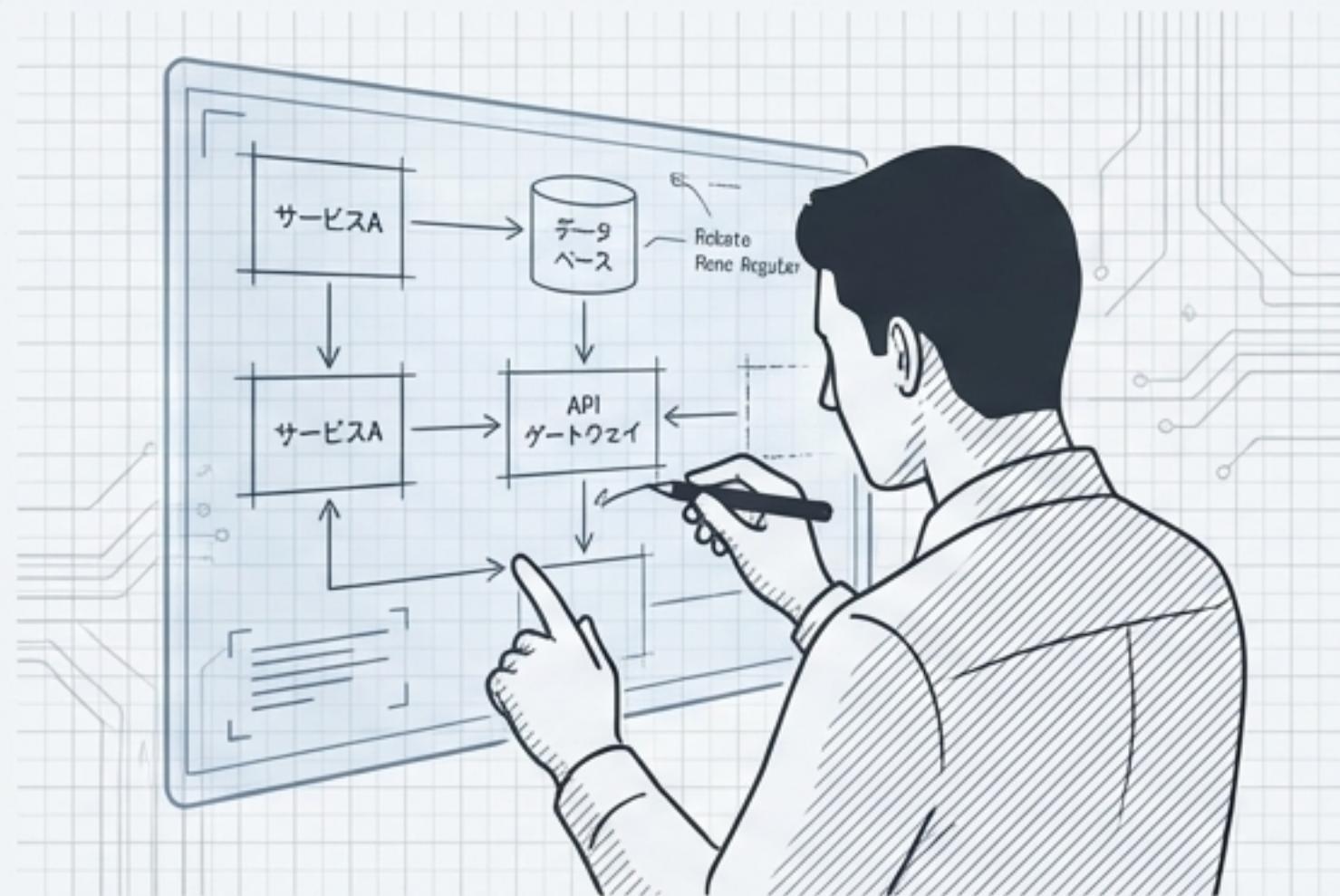
開発経験が個人に埋もれ、チームで共有されない。



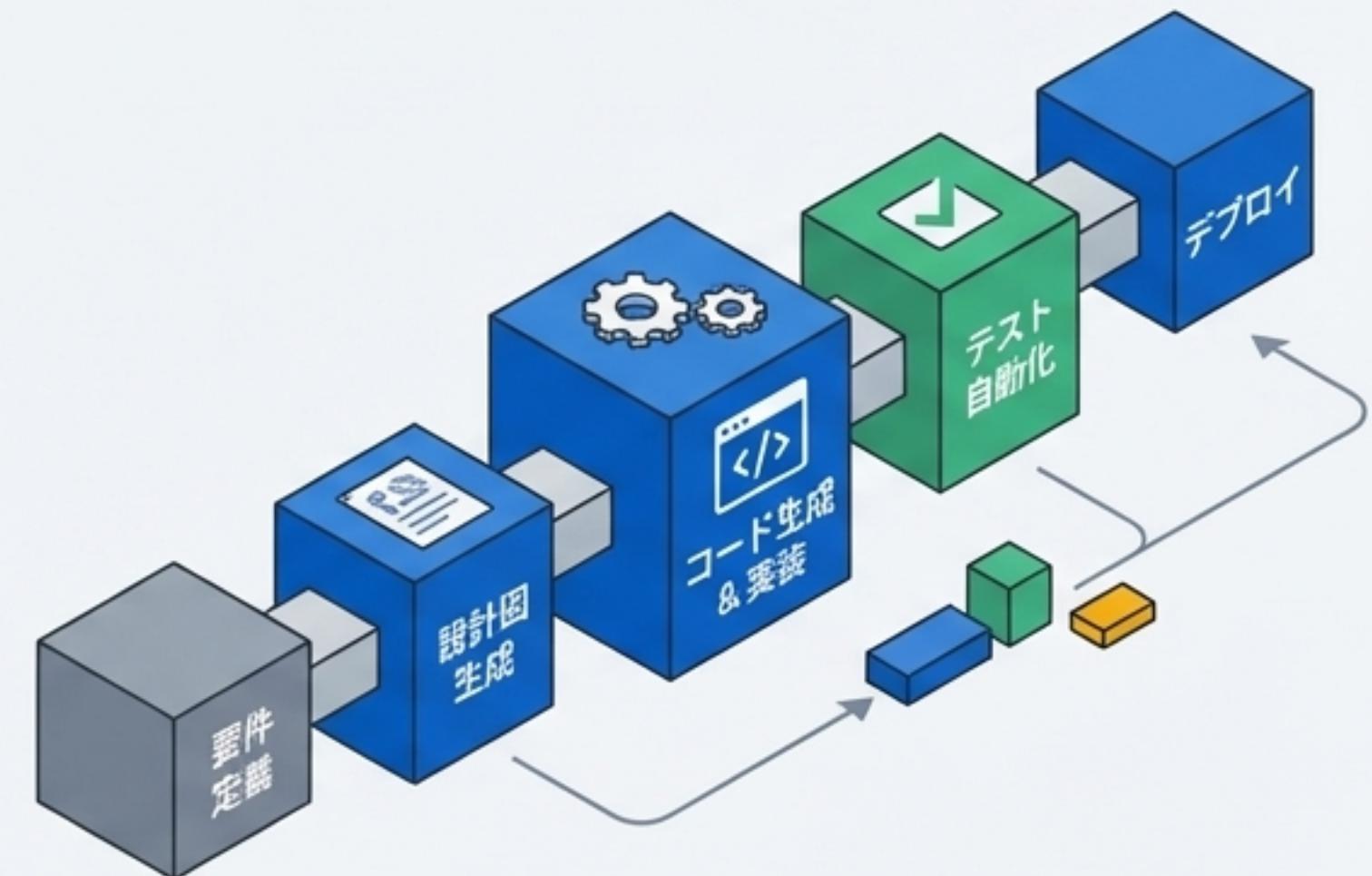
コンテキストスイッチ

高度な設計から実装の詳細へと頻繁に切り替えることによる精神的な負荷。

AI Factoryが実現する、新しい開発パラダイム

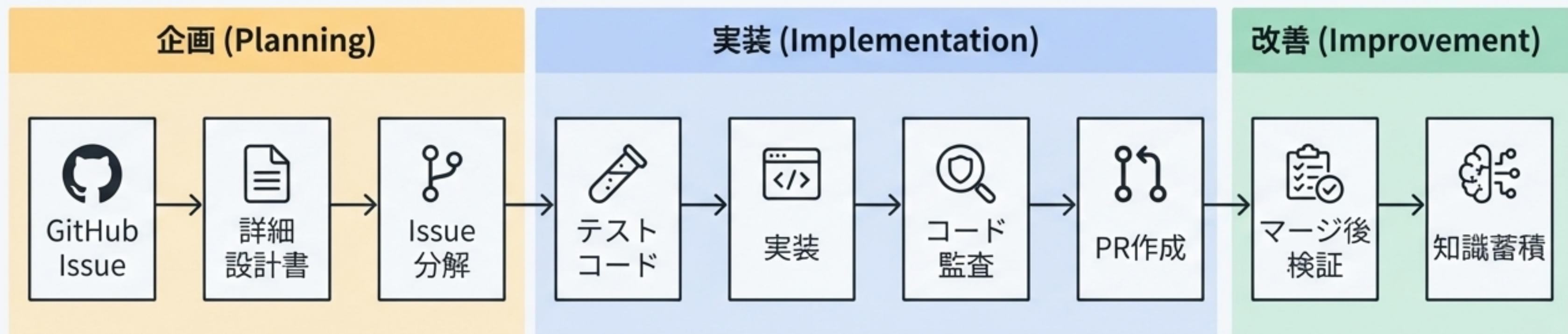


人間は「何を」作るかに集中する



AIは「どう」作るかを担当する

システム全体像：9つの専門AIエージェントによる協調開発



開発ドリームチーム：企画フェーズのエージェント



Product Manager
(@Product_Manager)

「要求の翻訳家」

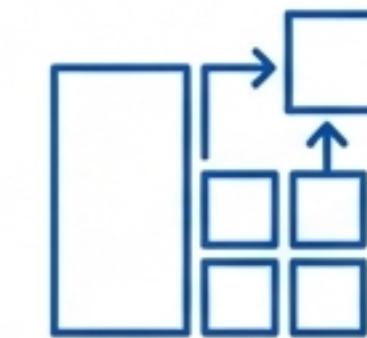
ユーザーの曖昧な要求を、
実行可能な技術仕様書に変換。



Critic (@Critic)

「厳格な設計レビュー」

仕様書の矛盾、リスク、曖昧さを徹底的に洗い出すニアアーキテクト。



Issue Planner
(@Issue_Planner)

「タスク分解のプロ」

承認された仕様書を、1日で完了できるサイズのGitHub Issueに分解。

開発ドリームチーム：実装フェーズのエージェント



QA Engineer (@QA_Engineer) 「品質の門番」

TDDを実践。実装前に網羅的なテストコードを設計・作成。



Coder (@Coder) 「熟練の実装者」

テストをパスし、規約に準拠した高品質なコードを記述。



Tech Lead (@Tech_Lead) 「コード監査官」

静的解析と変異テストで、コードとテストの品質を厳格に検証。



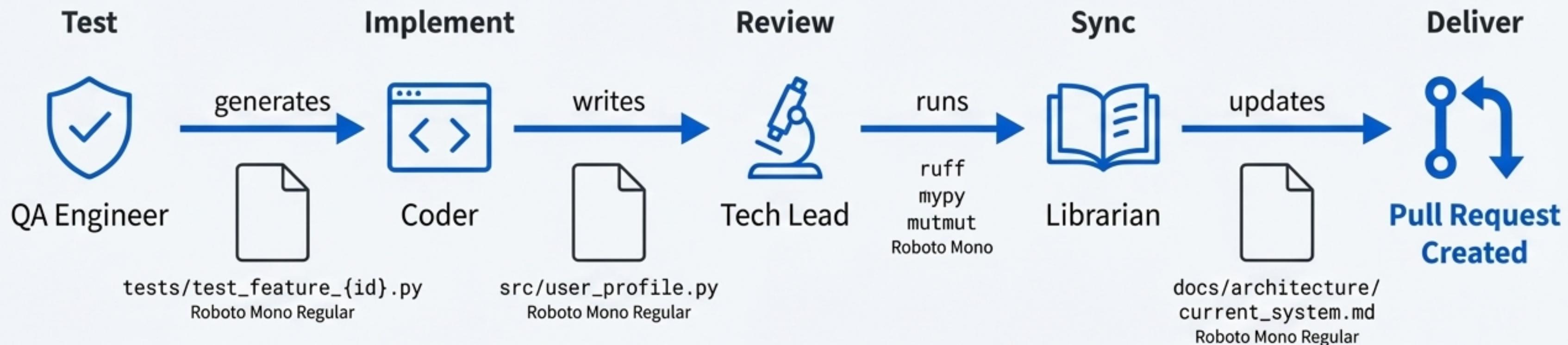
Librarian (@Librarian) 「ドキュメントの守護者」

コードの変更を検知し、システム全体のドキュメントを常に最新に保つ。

コアワークフロー：IssueからPRまでの一気通貫自動化

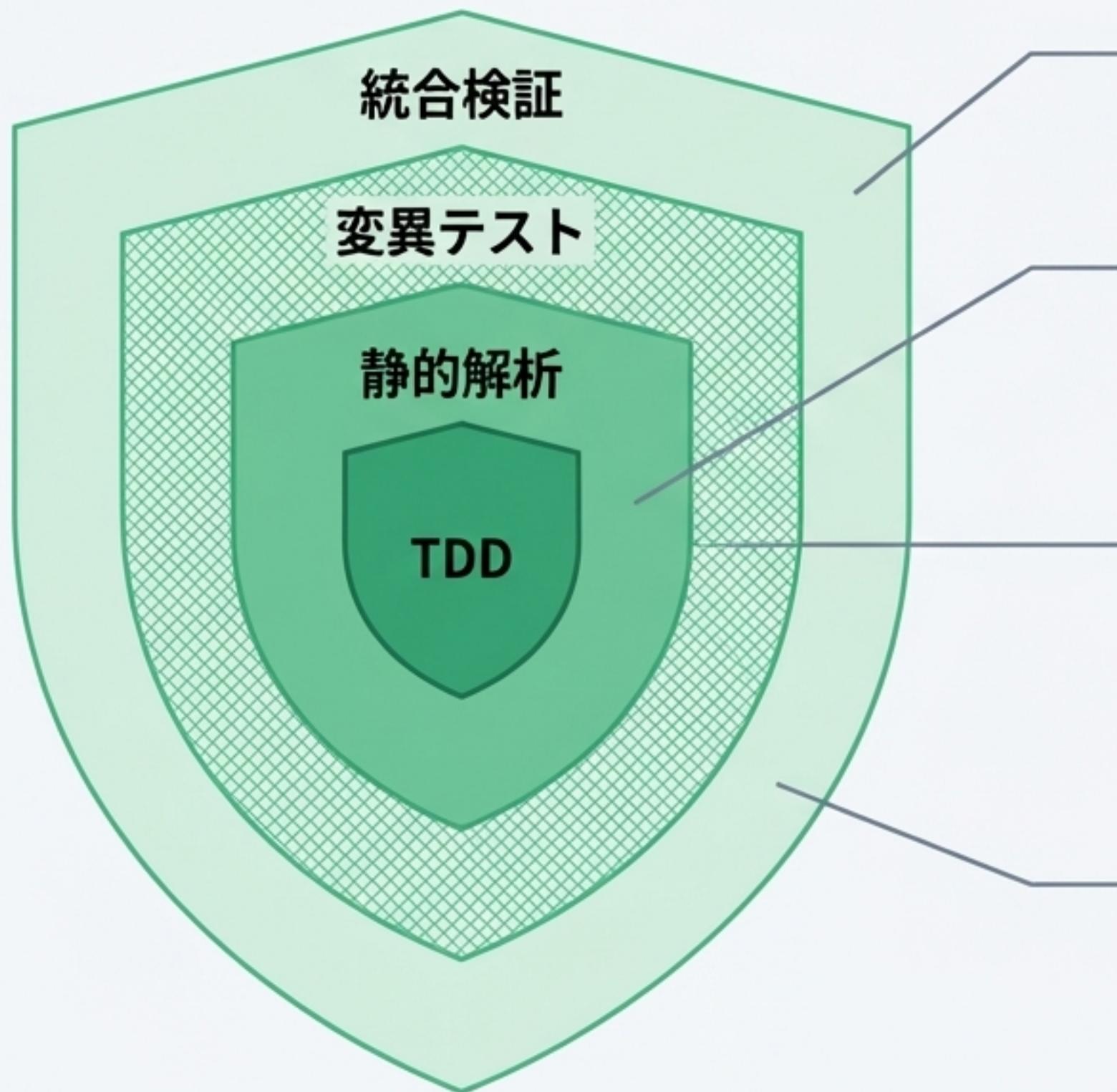
User Action: `gh issue create ...`

AI Factory Command: `/run {issue_id}`



このコマンドは完全自動実行モード（確認なし）で動作します。

第一の柱：妥協なき品質保証



Layer 1: TDD (Test-Driven Development)

QAが先にテストを書き、Coderがそれを満たす実装を行う。

Layer 2: 静的解析 (Static Analysis)

ruff (Linter) と mypy (Type Checking) でコードの健全性を機械的に検証。

Layer 3: 変異テスト (Mutation Testing)

mutmut を使用。コードをわずかに改変（変異）させ、既存のテストがその変更を検知（殺す）できるかを確認。「テストの品質」そのものをテストする。（Tech Lead rejects if score < 80%）

Layer 4: 統合検証 (Integration Validation)

マージ後、Validatorエージェントがシステム全体の動作を再検証。

第二の柱：自己改善システム「Kaizen」

記録 (Record)

各エージェントが作業中の試行錯誤や学びを「作業メモ」(`memos/issue-{id}-*.md`)に自動記録。

再利用 (Reuse)

抽出された知識を`claude.md`に蓄積。次回以降、全エージェントがこの知識を参照し、より賢く動作する。



蒸留 (Distill)

Scrum Masterエージェントが全メモを収集。Issue固有の情報を捨て、汎用的な知識を抽出。

Annotation

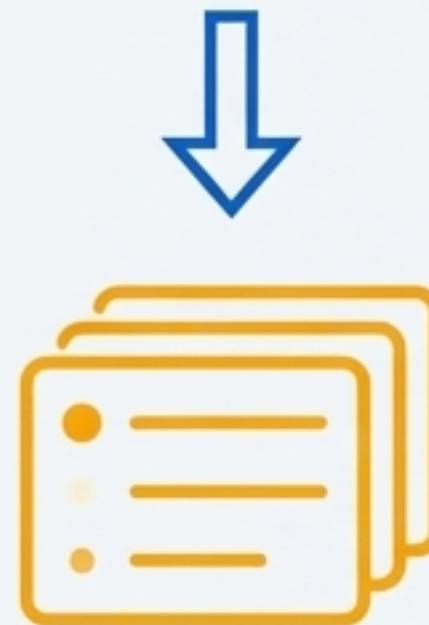
- ✖ 「Issue #123のバグを修正した」
- ✓ 「ライブラリXのバージョンY以降では○○に注意」

シンプルな使い方：2つのコマンドでプロジェクトが動く

企画 (Planning Phase)

`/prepare`

ユーザーと対話し要件をヒアリング後、仕様書作成 → レビュー → Issue分解までを完全自動実行。

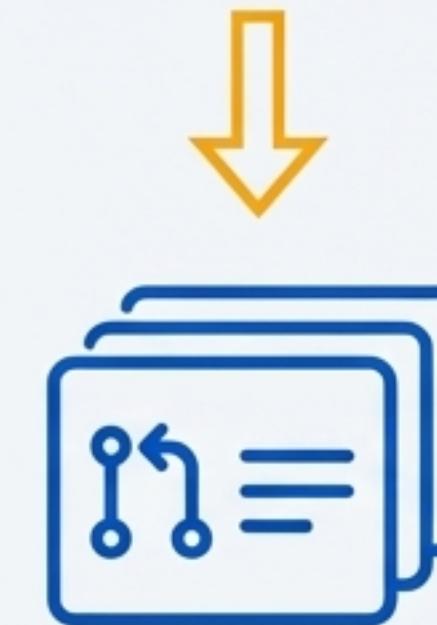


GitHub Issues

実装 (Implementation Phase)

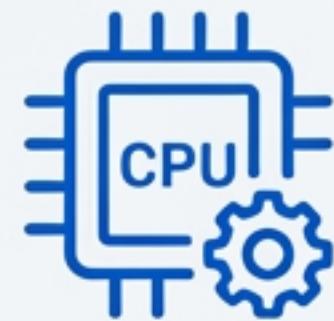
`/auto`

`label:todo` が付いた全Issueを検知し、若い番号順に自動処理を開始。テスト生成から PR作成まで確認なしで連続実行。



GitHub Pull Requests

AI Factoryを支える技術スタック



実行基盤 (Execution Engine)

Claude Code



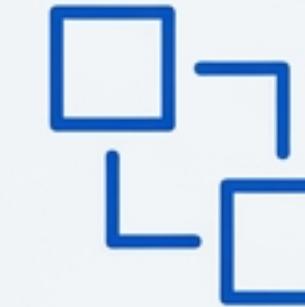
パッケージ管理 (Package Management)

uv (高速なPythonパッケージマネージャー)



品質保証 (Quality Assurance)

pytest, ruff, mypy, mutmut



インフラ連携 (Infrastructure Integration)

GitHub CLI (gh), Git Worktree

まとめと今後のビジョン

AI Factoryの現在

- ✓ 専門家チーム (Expert Team): 9体のAIエージェントが協調し、開発全工程をカバー。
- ✓ 高い品質保証 (Robust Quality): TDD、静的解析、変異テストによる多層防御。
- ✓ 自己改善 (Self-Improving): **Kaizen**機能により、開発経験を知識として蓄積・再利用。

今後のビジョン

