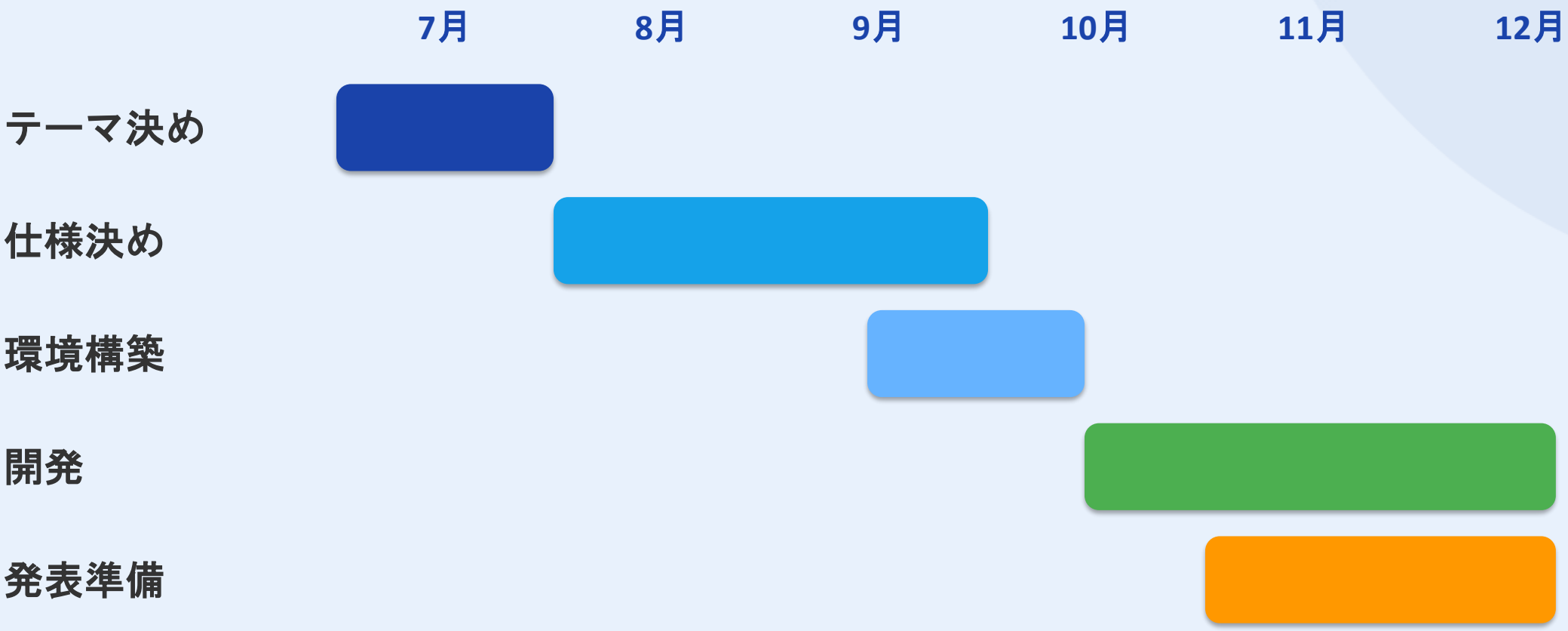
大分県情報サービス産業協会

# ～人＋AIで創る～
# プロダクト開発シミュレーション


大分シーイーシー株式会社
　　津野　聖矢
株式会社オーイーシー
　　久保田　将希
大銀コンピュータサービス株式会社
　　竹中　大志
東芝デジタルエンジニアリング株式会社
　　安東　聖七
モバイルクリエイト株式会社
　　熊谷　紅音
　　後藤　知美
SCSKニアショアシステムズ株式会社
　　秋枝　尚樹
　　志村　温子
　　山田　理矩

# 活動概要

## 活動方針：近い未来の開発プロセスを体験してみる



# 活動概要

| アドバイザー | 部会メンバ：9名 | | |
|---|---|---|---|
| | 部会長 | 副部会長 | |
| 安藤　淳禎 | 秋枝　尚樹 | 志村　温子 | 安東　聖七 |
| 谷口　泰生 | 久保田　将希 | 熊谷　紅音 | 後藤　知美 |
| 山本　一博 | 竹中　大志 | 津野　聖矢 | 山田　理矩 |

# はじめに

Devin

Claude Code

Cursor

## AIエージェント使ってますか？

Cline

Codex

Github Copilot

# はじめに

OISAの活動が始まって以降も、AIエージェントの発展とGPT5、Claude 4.1/4.5などの基盤モデルの登場による推論能力の向上によって、AIが自律的にタスクを実行し、プロジェクトを進行させる**"AI駆動開発"**の導入が現場レベルでも進み、広がりをみせてきています。

従来の「人がAIを使う」スタイルから、「人とAIが共に創る」スタイルへと、開発の在り方が大きく変わりつつあります。

私たちのチームは、この変化の最前線を体験することを目的に、AIエージェントを活用した

**プロダクト開発シミュレーション**を実施しました。

# 開発

### ▼TDD(テスト駆動開発)：特徴

・テストを先に作成し、小さな単位で段階的に実装を進めることで、仕様と実装の整合性を常に保つ。
・テストコードが仕様書として機能し、開発過程のドキュメントとしても活用できる。

### ▼TDD(テスト駆動開発)：採用理由

・AIが生成するコードの品質を検証しながら、安全にリファクタリングを行える開発環境を提供する。
・継続的な改善を通じて、保守性と信頼性の高いコードベースを安定的に維持できる。

# 開発

## ①エージェント型

AIが自ら計画を立て、自律的に行動し、複数のツールを活用しながらタスクを完了させる。

## ②ワークフロー統合型

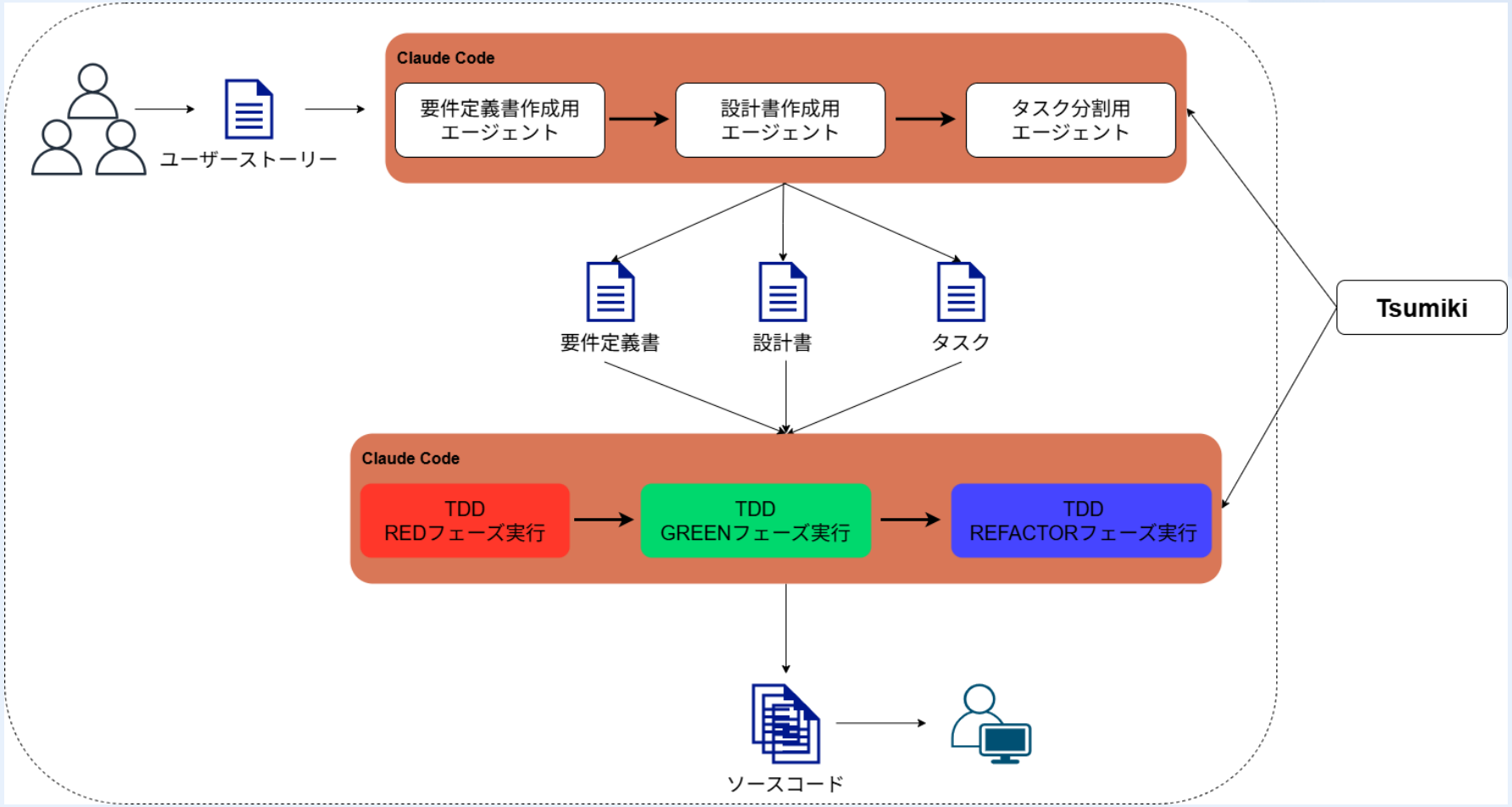
従来のCI/CDのパイプラインにAIエージェントを組み込み自動化を行う。

# 開発 ～エージェント型～

## ①エージェント型

# 開発 ～エージェント型～

## 各メンバーが以下の開発フローでAIエージェントを用いた開発を実践



# 開発 ～エージェント型～

## TDD REDフェーズ：失敗するテストを書く 🔴

### ▼ REDフェーズで作成する成果物

#### 🔴 失敗するテストコード

・期待する動作をテストコードで表現

・実装がまだ無いのでテストは必ず失敗する

・この「赤」が次のGREENフェーズのゴールになる

*※先にテストを書くから、何を作るべきか明確になる！*

# 開発 ～エージェント型～

## TDD GREENフェーズ：テストを通す最小限の実装

### ▼ GREENフェーズで作成する成果物

#### テストを通す最小限の実装コード

・テストが通ることだけを目標に実装

・コードの美しさより動作を優先

・「まず動く」状態を素早く作る

*※ まず動かすことを優先！早く方向性のズレに気づける*

# 開発 ～エージェント型～

## TDD REFACTORフェーズ：コードをキレイに

### ▼ REFACTORフェーズで作成する成果物

**リファクタリング済みのキレイなコード**

・重複コードの除去（DRY原則）

・命名の改善、関数の分割

・テストが通ることを確認しながら安全に整理

*※ テストがあるから的から外れない！どんどん改善できる*

# 開発アプリの概要

## 実装機能

### チャット機能
・ユーザが自由入力でキャラクターと対話

### パラメータ管理機能

・キャラクターの状態を0～500で数値化
・パラメータの組み合わせに応じて自動生成
・状態説明をフィードバック

# デモンストレーション

ここからは今回開発したアプリのデモンストレーションを行います。

# エージェント型実践してみて

## 良かった点

- 作業の自動化による効率化・作業負荷の軽減、改善提案等のサポート
- テスト先行の習慣化を支援
- テスト粒度の調整が容易
- コード品質の向上
- 自然言語でテスト仕様を作りやすい（Claude Code）
- フィードバックループが短い（Tsumiki）

# エージェント型実践してみて

## 苦労した点

・処理のブラックボックス化
　→全体としての把握が困難、説明責任が果たしづらい

・導入・運用の工夫が必須
　→高性能モデルが必須、ユーザストーリーの品質基準策定が必要

・フレームワーク依存が強くなる
　→保守性の低下、セキュリティリスクの潜在

・リファクタ耐性が低いテストが生成されることがある
　→適切な粒度でのテスト設計

# エージェント型実践してみて

## AIとの協働で実感したこと

- 機能設計とユーザーストーリーを細かく定義するほど、開発の方向性が明確になり成果が安定した
- AIによるテスト案の提示が、抜け漏れの防止や観点の広がりにつながった
- TDDの手順を踏むことで、作業全体が整理され品質とスピードの両方が向上した

# エージェント型実践してみて

## AIとの協働で実感したこと

- AIを使うことで作業の立ち上がりが早くなり、試行錯誤の時間が大幅に短縮された
- マークダウンで構造化して入力するだけで、AIの理解精度が大きく変化することを実感
- 人とAIの協調によって、普段気づけない観点に触れられるなど学習効果が大きかった

# エージェント型実践してみて

## 今後の展望

- 設計が明確に決まっている場合AIはその設計に従って効率的に実装

  → AI はその設計に従って効率的に実装できる

- 設計が不明瞭な場合、AIは推測しながら実装

  → 実装に入る前の設計段階がかなり重要

- 全てAIに任せた場合、全体把握が難しい

  → コマンドをただ実行するだけではなく、コマンドの意味や処理内容を理解しながら実行することが重要。

# エージェント型実践してみて

## 今後の展望

- 処理にかかる時間が想定より長くなる可能性がある

  → 余裕を持ったスケジュールを確保する

- 要件定義と実装仕様に差異が生じる場合がある

  → 小さな単位で実装させ、その都度内容を確認する

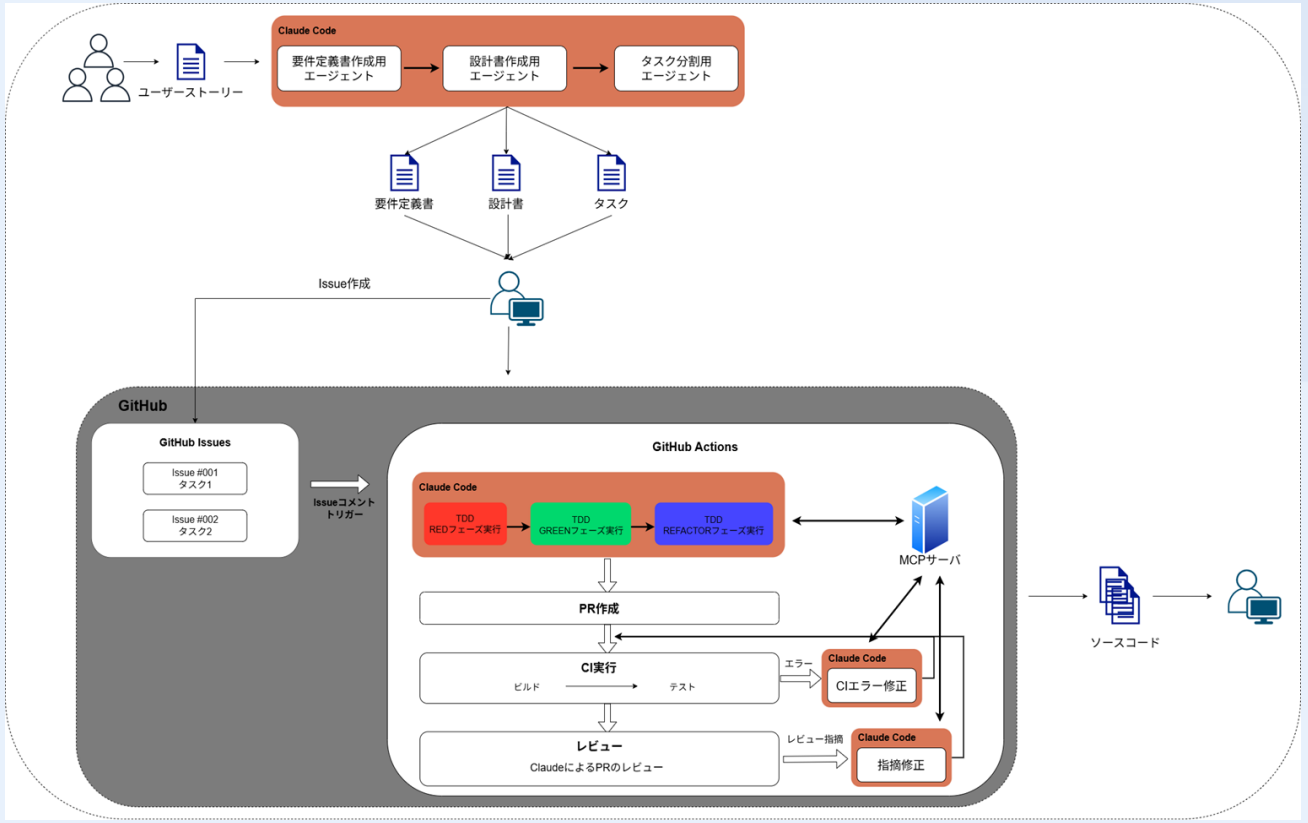
- レートリミットの影響で開発が遅延する可能性

  → プランの制限を把握し、別ツールの併用も検討する

# 開発プロセス自動化の実践

## 人が都度指示を出すのではなく、プロセスそのものを自動化

実装からレビューまでをシームレスに統合し、
エージェントの自立性を最大限に活かす。
従来のエージェント型とは異なり
ワークフロー全体が一つのシステムとして機能。



# Step1：ワークフロー起動

## タスクの指定

実装したい機能やバグ修正の内容を具体的に指示。
ここがプロセスの起点となり、明確なインプットが成功の鍵となる。

## Claude Code起動

ワークフロー内でAIエージェントが起動し、対象タスクの実装を自律的に開始。
人間は起動後の詳細な操作を行う必要はない。

# Step2：実装/CI/レビュー実行

### 自動実装

AIエージェントがコードを生成し、
プルリクエストを作成。

### CI / 自動テスト

実装後、即座にCIパイプラインが走り、
テストと静的解析を実行して品質をチェック。

### 自律修正

問題があれば、エラー内容をインプットにAIが自動で修正を行い、再度レビューへ。

# Step3：修正ループ

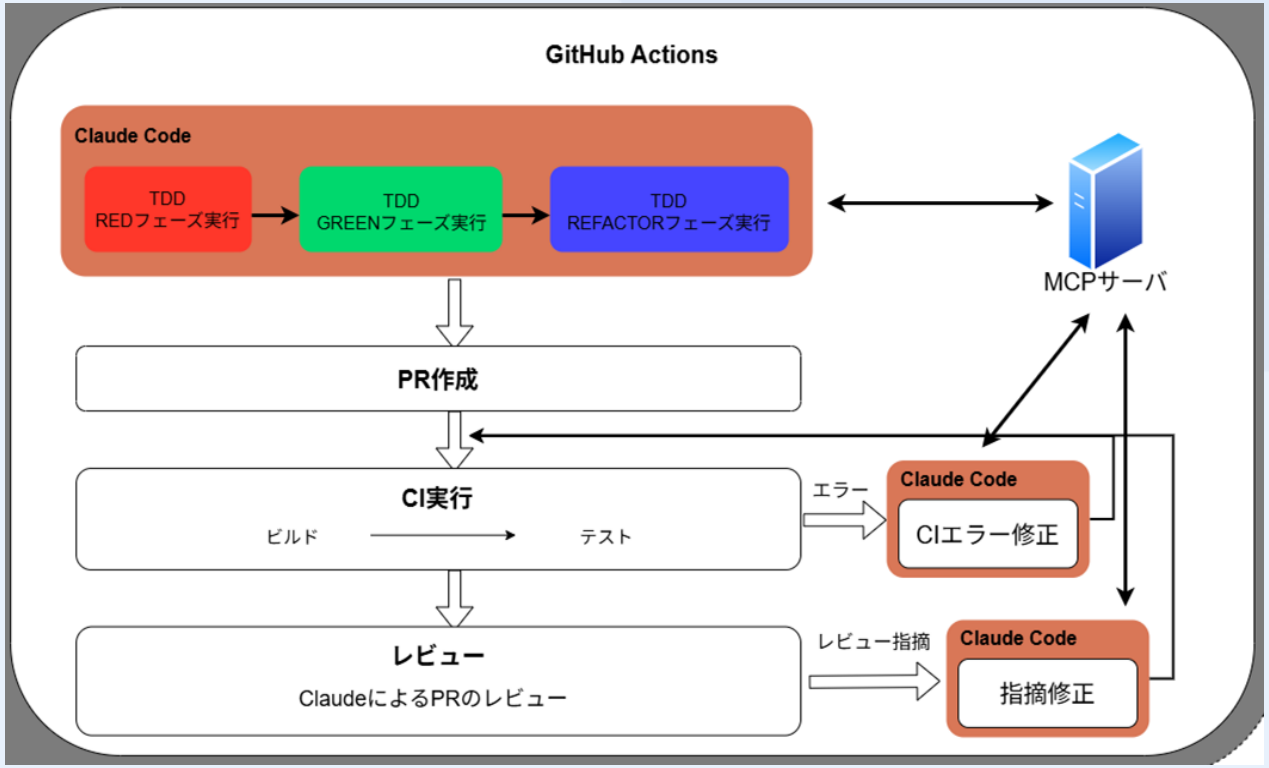
**承認されるまで継続する改善サイクル**

レビューがApproveになるまで、

修正と検証のサイクルを繰り返す。

このループにより、

一定の品質ラインまで自動で引き上げる。



# 実践における課題と解決策

## ⚠ 課題: 介入の難しさ

エージェント型と異なり、随時人間が介入して細かい指示を出すことが難しい。誤った方向に実装が進んでしまう可能性が高い。

## 解決策: ガードレール

エージェントを正しい方向に導くための「ガードレール」が重要。
仕様書、静的チェック、テスト
といった明確な境界線を設ける。

# 3つの重要なガードレール

## 1. 仕様書

**開発の方向性**

明確な要件定義が意思決定の指針となる。
使用技術を予め絞ることも有効。

## 2. 静的チェック

**品質の担保**

Lintや型チェックで規約違反を早期検出。
品質基準からの逸脱を未然に防ぐ。

## 3. テスト

**振る舞いの担保**

機能の正しさを検証。
期待される挙動から逸れた実装をブロックし、
修正を促す。

# テスト駆動開発の有効性

## ガードレールを機能させるアプローチ

- **方向性の明確化:** 仕様書 → テスト → 実装の流れで迷走を防ぎます。
- **品質の組み込み:** 実装前のテスト作成で、品質基準を先に定義します。
- **自己検証サイクル:** AIエージェントに最適なフィードバックループを提供。

# AIエージェント導入の効果

| 項目 | 効果 | 詳細 |
| --- | --- | --- |
| **開発速度** | 向上する ↗ | コード生成、テスト作成、ドキュメント作成等の自動化により、実装スピードは劇的に向上する。 |
| **リリース速度** | 必ず向上はしない ⚖ | すべてのプロジェクトで即座にリリース速度向上につながるわけではなく、解決すべき新たな課題が出現する。 |

# 解決すべき明確な課題

## 品質のばらつき

AIが一日に大量のPRを作成しても、品質が伴わなければ却下され、手戻りが発生する。量だけでなく質をどう担保するかが鍵になる。

## レビューのボトルネック

現状は人間によるレビューが必須。AIのPR作成速度に人間のレビューが追いつかず、タスクが滞留してしまう。

# 求められる環境

## 情報基盤の整備

仕様書やルールが常に最新化され、エージェントが体系的にアクセス可能な状態を維持する必要がある。

## AIフレンドリー

曖昧さのない明確な記述が求められる。ドキュメントのMarkdown化により、可読性を向上させる。

## 品質担保の仕組み

大量のPRを処理できるプロセスが必要。静的エラー自動修正や、明確な評価基準の策定が必要。

# 理想と現実のギャップを埋める

最初から完璧に環境を整備するのは困難

まずは「コストが低く」「効果を実感しやすい」

小さな一歩から始めることが重要

# まずはここから

## 開発ルールのマークダウン化から
## 始めてみませんか？