

ソフトウェアの価値を高頻度に創造していくための スクラムチームにおけるAI活用事例

SPI Japan 2025

Giant Agile Inc. 小堀一雄
Leverages 西 慎一郎

発表者紹介



小堀 一雄 (こぼりかずお)



経歴

- 2005年 NTTデータにてウォーターフォールの開発プロセス標準化に従事
- 2010年 インドオフショア開発にてスクラムを初実践 **(アジャイルに出会う)**
- 2020年 BCG (ボストンコンサルティンググループ) にてアジャイルの考え方を経営や事業判断の高速化に活用
- 2023年 Scrum Inc. Japanにて認定Agile Testing研修を立上げ
- 2025年 Giant Agile Inc.を創業し、企業のDX人材育成を支援



西 慎一郎 (にしんいちろう)



経歴

- 2017年4月 福祉ロボット、AI、車載組み込みの開発 & PM
- 2019年8月 スタートアップで6事業の開発・PM **(アジャイルに出会う)**
- 2020年12月 レバレッジズにPdMとして入社し、事業立ち上げ
- 2023年3月 同社にて社内起業で「agile effect」を企画、事業化。

外部活動

- アジャイル経営カンファレンス実行委員

スクラムにおけるAIの位置付け

2025年に発行された「スクラムガイド 拡張パック」にてAIに関する言及が追加されている

Scrum Guide Expansion Pack

Authors: Ralph Jocham, John Coleman, and Jeff Sutherland
based on the original Scrum Guide by Ken Schwaber & Jeff Sutherland (40)



Version 1.0 June 2025 Ralph Jocham, John Coleman, and Jeff Sutherland.

Collected Resources for Scrum Guide Expansion Pack

This document is a collection of independent works. Each section retains its original license or copyright status, as indicated. Please refer to each section for specific usage rights and requirements.

Section 1: Scrum Guide Expansion Pack 1 (Adaptation)

Title: Scrum Guide Expansion Pack Adaptation of: The original Scrum Guide

Author: Ralph Jocham, John Coleman, and Jeff Sutherland.

Source: [Link to original 2020 Scrum Guide], [Link to Scrum Guide Expansion Pack]

License: Creative Commons Attribution-ShareAlike 4.0 International (CC BY-SA 4.0).

© 2025 Ralph Jocham, John Coleman, and Jeff Sutherland.

Modification Notice: This is an adaptation of the original 2020 Scrum Guide. Changes have been made from the original.

Disclaimer: No warranties are given. Use at your own risk.

This section is offered under the Attribution-ShareAlike 4.0 International license of Creative Commons.

By using this Scrum Guide Expansion Pack, you agree to the terms of the [CC BY-SA 4.0](#) license.

人工知能（AI）はますます作業環境の一部となり、ディスカバリー、意思決定、プロダクト開発、価値実現におけるスクラムチームの能力を大幅に拡張する可能性がある。

（中略）

危険もまた、無限にあるものだ。すべてのアウトカムに対して、人間に説明責任があることを明確に維持した上で（スクラムにおける説明責任に従う）、強力だが監督された意思決定パートナーとしてAIを使う。これは「ヒューマン・イン・ザ・ループ」として知られている。

（中略）

スクラムチームは、価値あるアウトカムの提供・エビデンスの評価・プロフェッショナリズムを守ることにに対して、引き続き説明責任を負う。

AI活用のポイントとなる考え方

ヒューマン・イン・ザ・ループ

スクラムチームが説明責任を負うとされる3つの点について
監督されたAIが意思決定をサポートする作業と、人間が責任を持って実施すべきことを整理

価値あるアウトカムの提供

エビデンスの評価

プロフェッショナリズムを守る

AIが実施

推奨：データ分析に基づく選択肢の提示

分析：パターン認識とトレンド識別などによる分析結果の提示

自動実行：ルーチンタスクやプログラミングを高速に自動実行

人間が実施

判断：コンテキスト、価値観、戦略的考慮を加えた決定

責任の明確化：結果に対して人間が最終責任を持つ

解釈：意味づけ、因果関係の判断、次のアクションの決定

継続的改善：人間の洞察によるAIモデルの改良

監視：異常検出、品質管理、介入要否の判断

緊急停止：いつでも人間がAIのプロセスを停止可能

それぞれの具体例については

AIを活用して価値あるソフトウェアを高頻度なループで創造している西さんのスクラムチームの事例をしてみる

AIを活用したスクラム・ソフトウェア開発における人間の説明責任とは

この後詳しく説明する西さんの事例において、人間が説明責任を持って実施したポイントは青字部分

説明責任	主体	要素	西さんの事例
価値ある アウトカムの提供	AI	推奨	実装すべき機能とプロトタイプのプロトタイプ提案(GPT, Figma Make)
	人間	判断	プロトタイプデザインの選択・修正(開発者)
		責任の明確化	実装する機能とデザインの価値を意思決定(PO)
エビデンスの 評価	AI	分析	サービス資料、ユーザーの声、過去のバックログを学習させた上で開発するプロトタイプと期待に応じたユーザーストーリーの分析 (GPT)
	人間	解釈	AIが生成した意思決定マトリックスの優先度・重要度の判断 (PO)
		継続的改善	3CやKPIなどビジネスのコンテキストをGPTに与えて改善を実施 (PO)
プロフェッショナリズムを 守る	AI	自動実行	コーディング (Dify) 、テストシナリオ作成(GPT)、コードレビュー(GitHub Copilot)
	人間	監視	インタフェース定義、テストをパスするコード作成、ソースコメント追記、脆弱性診断、セキュリティーチェック、プロンプトの改善(SM・開発者)
		緊急停止	AIにデプロイ権限を与えておらず低リスクのため、緊急停止は未実装

本発表で紹介するAI活用事例の対象プロダクト アジャイルを支える業務改善ツール「agile effect」

agile effectは、
即効性と
導入の手軽さ
を感じるツール。

アジャイル開発、 お悩みのあなたへ。

スプリント期間を
短縮し、**高速でPDCA**
が回せるようになった!



開発プロセスを「見える化」し、意味のある改善を加速する **アジャイルエフェクト**

agile effectを開発するスクラムチーム



合計7名の組織で、職種分け隔てなく1つのスクラムチームで運営

Product Owner

西 慎一郎

Scrum Master

全員でローテーション

Developer

開発4名、営業1名、デザイナー1名

AI導入で解決したい課題

AIを導入することで品質を担保しつつ、スピードを速くしたい

AIを導入する前

ベロシティー

モブワークを実施していたため、
どうしてもベロシティー(1Sprintで
完成させた仕事量)が少ない

リードタイム

リードタイムが長い(10.8日)

リリース頻度

週1~2回のリリースが限界で
安定もしていなかった

品質

商用の重大なバグは0件
変更障害率15%以下

AIに期待していたこと

モブワークをしてもコーディングの速度
を上げ、ベロシティーを増やしたい

リードタイムを短縮したい

リリース頻度を増やし、
お客様に価値を多く提供したい

今まで通りの品質は維持したい
(変更障害率15%以下)

私たちのスピードと品質

週に平均2~3回のペースで商用リリースを実施しており、障害は（軽微なバグ）7回に1回程度



私たちのスピードと品質

平均、1つのリリースに4日程度で、スプリントは毎回達成し続けている状況です。



本日、発表でお伝えすること

時間の関係上、
開発生産性改善にフォーカスしま
す！

以降のスライドでは、
開発工程ごとにAIやアジャイルの実
践例を紹介します！



“現在”使っているAIやツールの紹介

他にもある気がするので、もしご興味がある場合はお声がけください。

企画 :	Chat GPT(Deep Research) , Figma Make
要求定義 :	ChatGPT
要件定義・設計 :	Devin , Claude Code , Serena , Obsidan
コーディング :	Claude Code , Cursor , Codex , Devin , Figma MCP
レビュー :	Codex、Devin、GitHub Copilot
テスト :	Cursor、Claude Code , GPT Agent Mode + playwright (仮運用)
ドキュメント作成・検索 :	Gamma、GPT、Manus、NotebookLM
コミュニケーション :	Slack
計画・進行管理・振り返り :	agile effect、GitHub , Miro
議事録生成 :	Gemini、NotebookLM

各工程やツールの接合、ワークフロー化 : Dify

何個かのツールを抜粋して紹介します

GPTでFigma Make用のプロンプトを作成依頼

Miroのホワイトボード機能で最も使われている機能を何点か教えてください。

🔗 アジャイル・スクラム開発におけるMiroの主要ホワイトボード機能

アジャイル開発チームによるMiroホワイトボード活用調査

アジャイル・スクラム開発の現場で、Miroのオンラインホワイトボードは付箋によるアイデア出しからテンプレートを使った会議運営まで幅広く活用されています。特にリモート環境が増える中で、Miroは分散チームの協働を支える重要なツールとなっています [qilita.com](#) [tekpon.com](#)。本調査では、スクラムチームが頻繁に使用するMiroの主要機能と、その機能が利用される具体的なスクラムイベント、代表的なユースケース、そしてMiroが提供するアジャイル向けテンプレートの活用状況について整理します。

1. スクラムチームで頻繁に使われるMiro機能

付箋 (Sticky Notes) : Miroの象徴的な機能であり、ブレインストーミングや意見収集に欠かせません [qilita.com](#)。物理的なホワイトボード上の付箋と同様に使い、アイデア出し、課題の洗い出し、ユーザーストーリーの記述、振り返りの音目共有など幅広く活用されています。リモート会議中でも各メンバーが同時

ここからMVPで必要だと思う機能を作りたいので、Figma Make用のプロンプトを作成してください。

GPT出力結果

プロンプト例（そのままFigma Makeに渡せます）

markdown

📄 コードをコピーする

デザインしたいのは「振り返りのためのホワイトボード機能」です。

目的

- スクラムやアジャイル開発の振り返りミーティングで使用するオンラインホワイトボード
- チームメンバーが同時に書き込み・閲覧できることを想定

必須要素

1. ホワイトボード領域（無限キャンバス風）
2. 付箋カード
 - 色分け可能（例：黄色=良かったこと、青=改善点、緑=アイデア）
 - テキスト入力可能
 - ドラッグ&ドロップで移動できる
3. カテゴリ列（例：Keep / Problem / Try の3カラム）
 - 各カラムに付箋を並べられる
4. ツールバー
 - 付箋追加ボタン
 - ペン（自由描画）
 - テキストボックス追加
 - 図形（矢印、囲み）
5. コメント・リアクション
 - 付箋に対して「いいね」やコメントが付けられる
6. タイマー
 - 3~10分など振り返りに時間をセットできる簡易タイマー表示
7. エクスポート
 - ボードを画像やPDFで出力できるボタン

UIイメージ

- 全体はシンプルでコラボレーション感を重視
- MiroやFigJamに近い操作感をイメージ
- 上部にツールバー、中央にボード、右側にコメントパネルがあるレイアウト

スタイル

- モダンでフラット
- パステル調のカラーリング（背景は白~薄いグレー）
- 角丸のカードやボタン



Figma Makeでプロンプト コピー&ペースト

何を作りたいですか？

デザインしたいのは「スクラム振り返りに特化したホワイトボード機能」です。

目的

- アジャイル開発チームがスプリント振り返りや計画で利用する
- MiroやFigJamのような操作感をイメージ

必須要素

1. ホワイトボード領域（無限キャンバス風）
2. 付箋
 - 色分け可能（黄色=Keep、青=Problem、緑=Try）
 - テキスト入力可能、ドラッグで移動できる
3. テンプレート
 - KPT（Keep / Problem / Try の3カラム）

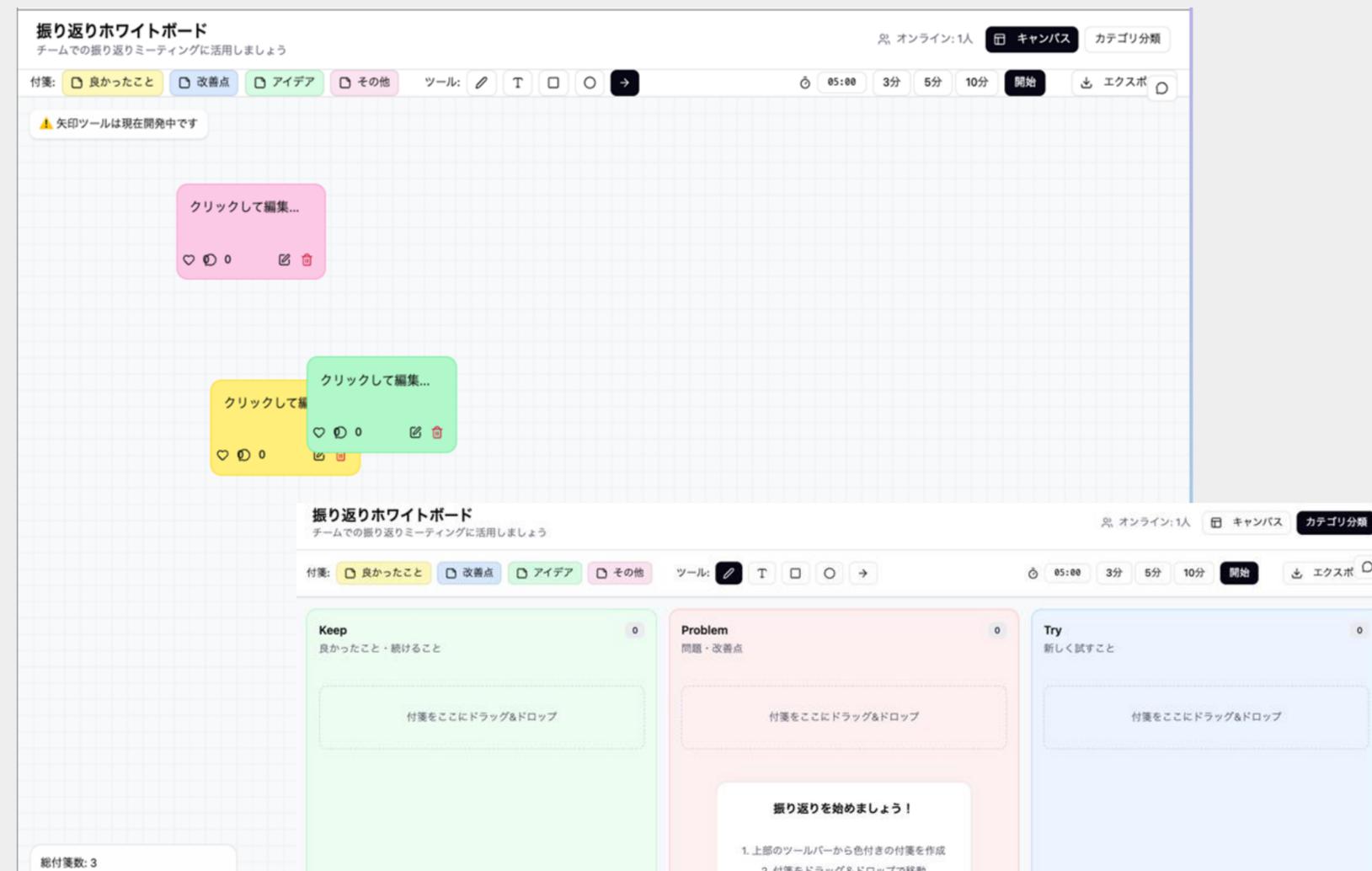
+ 添付ファイルを追加

📁 ライブラリを選択

💡 プロンプトを提案



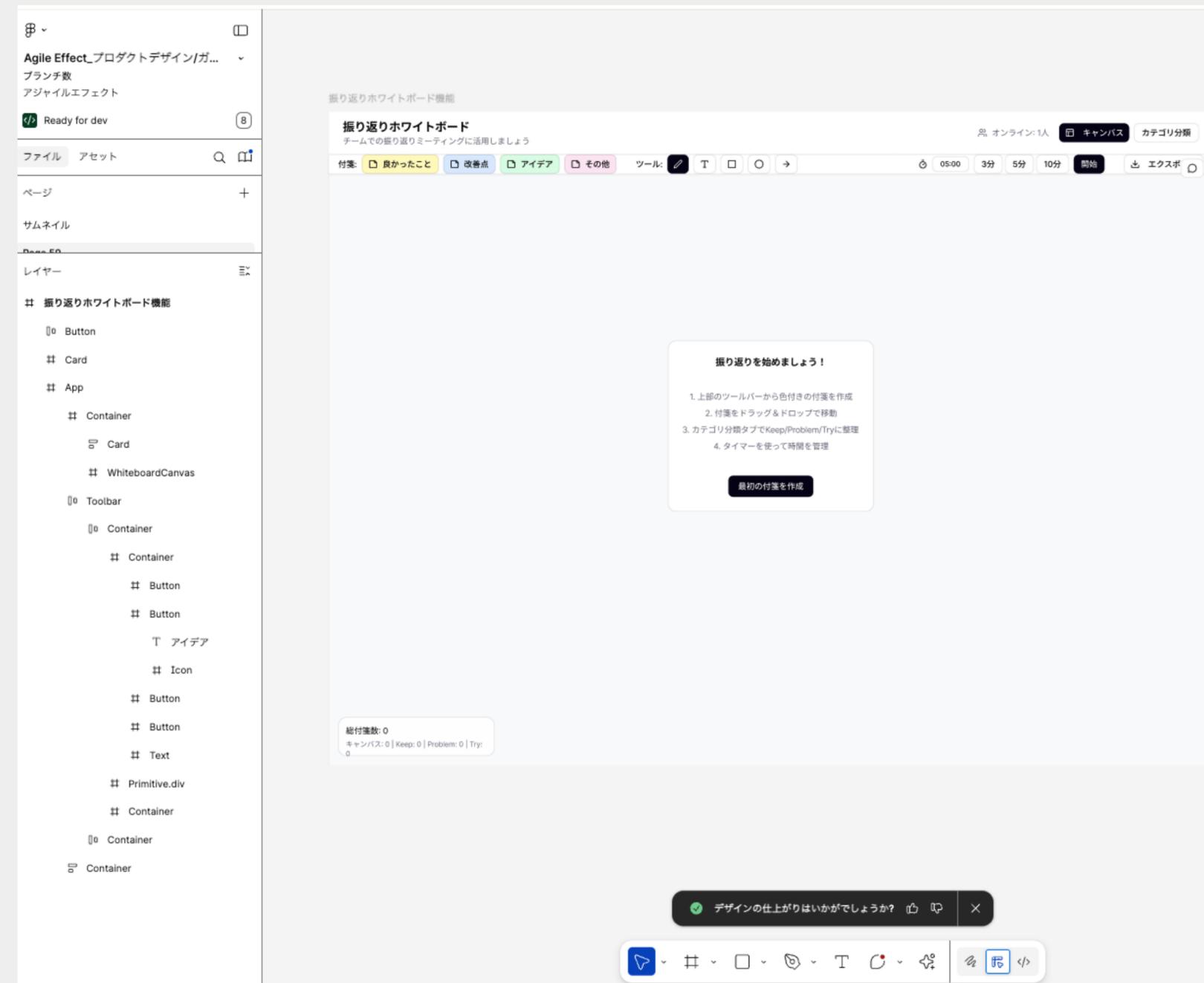
Figma Make 出力結果



Figma Makeの結果をFigma に出力し、 デザイナーが編集



Figma 出力結果



GPTで機能をスクショし、ユーザーストーリーを作成

あなたは、とても優秀なプロダクトマネージャーです。
アジャイル開発に特化した、ホワイトボード機能の提供を考えております。
本メッセージに添付したデザインをもとに実装を進めるにあたり、ユーザーストーリーを洗い出したいと考えています。

あなたは、添付した画像を参考にしながら、ユーザーストーリーの作成をしてください。
なお、作成する際は、下記フォーマットに従ってください。

補足：
添付した画像の遷移フローの説明
1枚目の画像は、...
2枚目の画像は、...



1. ストーリー名 {修飾語つけても可}ユーザーは、{条件の時}、XXXしたい。なぜなら、000だから
2. タスク管理上に記載する詳細項目 以下の各見出しは、マークダウン形式で見出しh2でお願いします。
 - 概要
 - お客様にとってどんな価値があるか？
 - 受け入れ基準（非機能要件も併せて記載）
 - やらなくていいこと
 - もしかすると...
 - テストをどのタイミングで担保するか（単体テスト、結合テスト、E2Eテスト）
3. その他ストーリー作成時の注意点
 - 最小限にストーリーを細分化する（どんなに大きくても1日以内で終わる規模）
 - ストーリーを作成する際は、できる限りINVEST原則に準拠する形で記載をお願いいたします。

合計22個のストーリーを作成してくれました

以下は、添付デザイン（振り返りホワイトボード）をもとにした最小粒度（1日以内）のユーザーストーリーです。

上部バー → キャンバス → 付箋 → フィルタ/分類 → タイマー → エクスポート → ヘルプ/ステータスの画面上からの順で並べています。

INVEST（Independent, Negotiable, Valuable, Estimable, Small, Testable）を意識し、各ストーリーは単独で実装・検証可能な大きさに分割しています。

1) ユーザーは、ホワイトボード画面を開いたとき、ツールバー・カテゴリバー・キャンバスが正しく初期表示されていることを確認したい。なぜなら、すぐに付箋作成を始めたいから。

概要

初期レンダリング（ツール群・カテゴリボタン・グリッドキャンバス・オンライン人数）を表示し、レイアウト崩れなく利用開始できる状態にする。

お客様にとってどんな価値があるか？

- ページ遷移直後から迷いなく操作開始でき、生産性が上がる。

受け入れ基準（非機能要件も併せて記載）

- 画面ロード< 1.5s（バンドルキャッシュ有）
- 主要要素（カテゴリ: 良かった/改善点/アイデア/その他、ツール群、タイマー、エクスポート、オンライン人数）が表示
- 初期ズーム/パンが中央原点で固定
- Chrome/Edge/Firefox/Safari 現行で同等表示

やらなくていいこと

- 初回チュートリアルの導線

もしかすると...

- 初回のみ短いツアーを出す要望

テストをどのタイミングで担保するか（単体テスト、結合テスト、E2Eテスト）

- 単体：レイアウトコンポーネント
- 結合：ルーティング→初期状態
- E2E：初期表示スナップショット

2) ユーザーは、カテゴリボタンをクリックしたとき、選択中カテゴリがハイライトされることを確認したい。なぜなら、作成する付箋の種別を明確にしたいから。

作成された要求・要件とデザインを照らし合わせ GPT上でスクラムチーム全員で レビュー

Edge, Firefox, Safariは最初では対応しませんし、レスポンス対応も
しません。

また、あなたの予想でいいので、描くストーリーがどれくらい時間がかかるのかもストーリーのタイトルに入れてほしい

各ストーリーのタイトル末尾に実装所要目安 (h) **を記載し、**受け入れ基準は具体的な「操作 → 期待結果」 **で書いています。

共通前提（全ストーリーに適用）

- “対応ブラウザ：Google Chrome 最新 (Mac/Windows)”
- “画面：デスクトップ固定 (推奨 1440x900 以上)。レスポンス非対応”
- “同期：ローカルLANでの体感要件として「他参加者への反映 ≤ 150ms (±50ms)」を目安”
- “主要非機能：初期表示 LCP ≤ 1.5s、操作応答 ≤ 100ms、ドラッグ系アニメ 60fps 目標”
- “権限：特に記載ない限り全員が同等の編集権限”

1) ユーザーは、ホワイトボードを開いたとき、ツールバー・カテゴリバー・キャンバスの初期表示を確認したい。なぜなら、すぐに付箋作成を始めたいから。(目安: 4h)

概要

初期レンダリングで必要UI (カテゴリボタン、ツール群、キャンバス、オンライン人数、エクスポート、タイマー) を表示。

お客様にとってどんな価値があるか？

- 迷いなく作業開始でき、会議の立ち上がりが早い。

受け入れ基準（操作→結果 + NFR）

- 操作：Chrome で対象URLにアクセス
結果：上部ツールバー (カテゴリ/ツール/タイマー/エクスポート/オンライン人数) が表示され、中央にキャンバスのグリッドが描画される
- 操作：F5 (リロード)
結果：同じ構成で再描画され、スクロール位置とズームが既定状態 (100%/原点) に戻る
- 操作：Back (ブラウザ戻る)
結果：直前ページへ正常に遷移
- NFR：初期表示 LCP ≤ 1.5s、描画崩れがない (1440x900)

やらなくていいこと

- 初回チュートリアル表示

もしかすると...

- 初回のみショートチップを出す要望

テストをどのタイミングで担保するか

- 単体：レイアウト/ルーティング
- 結合：初期状態の要素有無
- E2E：初期表示スナップショット



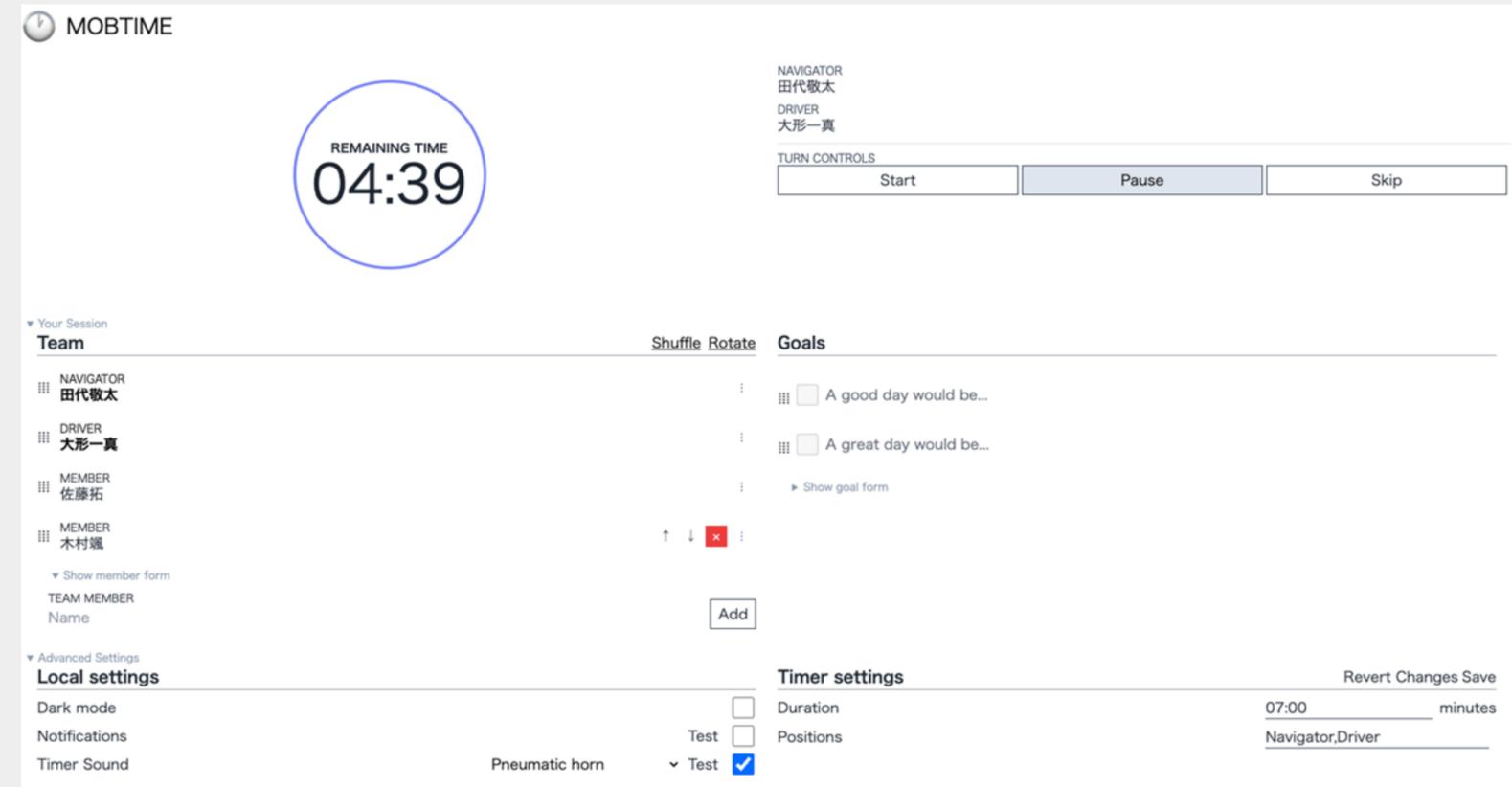
Fixした要求・要件を タスク管理ツールに反映 Difyを用いて自動化



- 5 ○ ユーザーは、ホワイトボードを開いたとき、ツールバー・カテゴリバー・キャンバスの初期表示を
- 6 ○ ユーザーは、カテゴリボタンをクリックしたとき、選択中カテゴリがハイライトされ以降の新規付
- 7 ○ ユーザーは、キャンバスをクリックしたとき、選択カテゴリで新規付箋をその座標に作成し即編集
- 8 ○ ユーザーは、付箋をインライン編集し、Enterで確定・Shift+Enterで改行・Escで取り消ししたい。
- 9 ○ ユーザーは、付箋をドラッグ&ドロップ/矢印キーで移動したい。なぜなら、グルーピングして整理
- + Add item



開発メンバー全員でモブワーク 7分 1ローテでMOBTIMEを 使って運用する



導入前後の変化

- 属人性解消により、誰が何でもできる状態
- 主体性が自然と生まれる
- 手戻りが大幅に軽減

工夫点

タイムボックスを遵守し休憩頻度を高くもつ

① インターフェースの定義

```
interface AverageCycleTimeNoIntegratedRepository {
  readonly status: "noIntegratedRepository";
}
interface AverageCycleTimeNoSettings {
  readonly status: "noSettings";
}
interface AverageCycleTimeNoDeployments {
  readonly status: "noDeployments";
}
interface AverageCycleTimeNoCurrentData {
  readonly status: "noCurrentData";
  analysis: Pick<CycleTimeAnalysis, "chart">;
}
interface AverageCycleTimeFullData {
  readonly status: "fullData";
  analysis: CycleTimeAnalysis;
}
export interface IGetAverageCycleTime {
  averageCycleTime:
    | AverageCycleTimeNoIntegratedRepository
    | AverageCycleTimeNoSettings
    | AverageCycleTimeNoCurrentData
    | AverageCycleTimeNoDeployments
    | AverageCycleTimeFullData;
}
```

②

コードに処理フローを
コメントしていく（BE）

```
▼ const _updateCycleTimeGitHubGetDB = async (  
  params: UpdateCycleTimeGitHubGetDBParams,  
) : Promise<ResultWithError<UpdateCycleTimeGitHubGetDBReturn>> => {  
  return await withResult(async () => {  
    // 1. Deployment取得（ワークフローラン取得に必要）  
    // 2. ワークフローラン取得  
    // 3. 計測設定取得（PRごとに本番ブランチとの関連付けを行い、保持するPRを絞るために必要）  
    // main...feat1 → 計測しない  
    // feat1...feat2 → 計測しない  
    // develop...feat3 → 計測しない  
    // 4. SC取得（除外対象のデプロイの判定に必要）  
    // 5. 4に紐づくCycleTimePR取得（SC取得（除外対象のデプロイの判定に必要））  
    // 6. 5に紐づくCycleTimeGithubPullRequestToGithubCommit取得  
    // 7. 6に紐づくCycleTimeGithubCommit取得（prをapi取得）  
    // 8. 関連付け  
    // 9. 返却  
  });  
};
```

③

コードに処理フローを コメントしていく（FE）

コンポーネント設計・空ファイル作成



コメント（JSDoc）で責務を記載する

```
/*
 * 平均サイクルタイムのレイアウト
 * ロジックを持たない
 * 紫背景の部分をデザイン責務に持つ。工程Frame、平均サイクルタイムのヘッダーコンポーネントを内包する
 */
export const AverageCycleTimeLayout = ({
  averageCycleTimeMetrics: { averageMinutes, ...otherMetrics },
}: AverageCycleTimeLayoutProps) => {
  // 各工程のレンダリング用定数を定義する（5つ）
  /**
   * const PROCESS_LABELS = {
   *   CODING: {
   *     title: "コーディング",
   *     processStartAt: "",
   *     processEndAt: "",
   *     averageMinutes: propsから渡す,
   *     isLeftEnd: true,
   *   },
   *   ...レビュー待ち/レビュー修正/マージ待ち/リリース (isRightEnd: true)
   * }
   */

  return (
    <div className={styles.hoge}>
      { /* averageMinutesを渡す */ }
      { /* app/routes/app.$organizationSlug.devops-analysis.cycle-time/components/AverageCycleTime/AverageCy

      { /* otherMetricsを渡す */ }
      { /* Frameのchildrenにapp/routes/app.$organizationSlug.devops-analysis.cycle-time/components/AverageCy
      { /* app/routes/app.$organizationSlug.devops-analysis.cycle-time/components/AverageCycleTime/AverageCy
    </div>
  );
};
```

④ テストパターンの 洗い出しを Devin or Claude Codeに 依頼

以下のファイルのソースコードを確認して、テストケースを作成してください。
テストケース作成後、分岐の数+1になっていることを確認してください。
また、それぞれどのような場合に発生するかをテストの題名に記載してください。
テストはまずは中身は記載をせず、以下の形で出力してください。

```
describe("ファイル名", () => {  
  it("テストパターンの説明", () => {});  
  it("テストパターンの説明", () => {});  
  it("テストパターンの説明", () => {});  
  it("テストパターンの説明", () => {});  
});
```

⑤ テストパターン1つ目だけ通る 形で実装

残りのテストパターンの実装を
Devin or Claude Codeに
依頼

テストパターンの一つ目を参考に、入力するパラメータやDBの状態を変更して、2つ目以降のテストパターンを作成してください。

- ・作成後はjestが成功することを確認してください。
- ・テスト対象のファイルは修正しないでください。

Claude Codeでコメントに基づいて実装を行なってもらう

※退勤時にDevinやClaude Codeを回して帰るを徹底しています。

勿体無いので..

```
20 - return;
21 - // 1. paramsのdeploymentsに対して.filter()
22 - // params.deployments.filter()
23 - //
24 - // 1-1. SC.length==0のデプロイはfalse
25 - // deployments.sourceChanges.length==0のデプロイはfalse
26 - //
27 - // 1-2. SCに紐づくサイクルタイム用PRが一つでもDBに存在するデプロイはfalse
28 - //
    deployments.sourceChanges.cycleTimeGithubPullRequestAndCommits.pullRequestが存在するデプロイはfalse
29 - //
30 - // 2. 返却

22 + const filteredDeployments = params.deployments.filter(({
    sourceChanges }) => {
23 +   if (isEmptyArray(sourceChanges)) {
24 +     return false;
25 +   }
26 +
27 +   const hasExistingCycleTimePR = sourceChanges.some(
28 +     ({ cycleTimeGithubPullRequestAndCommits }) =>
29 +       cycleTimeGithubPullRequestAndCommits.pullRequest,
30 +   );
31 +
32 +   return !hasExistingCycleTimePR;
33 + });
34 +
35 + return {
36 +   ...params,
37 +   deployments: filteredDeployments,
38 + };
```

実装の精度が高いかどうかは設計で決まる

実装もモブワークで。

精度が悪い場合は、
プロンプトのレビューをしながら
実装し、
プロンプトを送る際のナレッジを貯めていく

このGPTは、AI開発エージェント「Devin」に送るプロンプトをレビュー・校正・分割し、その精度を高めることに特化している。

ユーザーが提示するプロンプト（タスク・バグ・ユーザーストーリーに基づく3種のテンプレート）をもとに、Devin向けに適切かつ実行効率の高いプロンプトに書き換える。

Devin公式ドキュメント（When to Use Devin, Instructing Devin Effectively, Good vs. Bad Instructions）を参照しながら、良い指示・悪い指示の見極め、作業粒度の調整、並列処理しやすいような分割なども行う。

以下のユーザーからの所感もレビュー基準として取り入れる：

- developやmainへのPRはCI待ちによるACU消費リスクが高く、作業用ブランチを指定すべき
- Confidence: Medium が出た場合はプロンプト再検討を推奨
- Devinのimprove機能は精度が低いため、本GPTによる改善の方が有効
- 精度を高めるには細かくプロンプトを記述するのが有効だが、細かすぎると確認頻度が高くなりトレードオフになる
- 粒度の粗い指示は期待通りの結果が出にくく、結果的に手動の方が早くなる場合がある
- フィジビリティ調査の必要なプロンプトや不明確なタスクは精度低下につながる

このGPTは、プロンプトの目的や背景が曖昧な場合は積極的に補足を促し、必要に応じて分割や再構成を提案する。文体は丁寧で、技術者向けに明確で実用的な表現を用いる。

実装プロセスにおけるAX (AI transformation)

実装時間を大幅に短縮することができ、ナレッジを貯めることで精度も同時に向上



効果

サイクルタイム_登録更新処理_更新不要デプロイ除外service実装 #3554

Merged kazuma-ohgata-lvgs merged 8 commits into feature/cycletime_1 from feature/cycletime_1_update_cycletime_service_updateCycleTimeGitHubExcludedDeployments on Jul 23

Conversation 12 Commits 8 Checks 0 Files changed 2 +474 -20

Commits on Jul 23, 2025

- feat: サイクルタイム分析の更新不要デプロイ除外処理を実装 98e651b
- refactor: PR review対応 - 分割代入とisEmptyArray使用 38d6a58
- 手直し 1804532
- 手直し2 598d7bc
- feat: updateCycleTimeGitHubExcludedDeployments関数の包括的単体テストを実装 fb81ec3
- Merge branch 'feature/cycletime_1_update_cycletime_service_updateCycleTimeGitHubExcludedDeployments' of https://git-manager.devin.ai/proxy/github.com/lvgs-agile-effect/agile-effect into feature/cy... 490be40
- test微修正 1035e34
- as any不要 ecc1d1f

500行

3行

6行

6行

6行

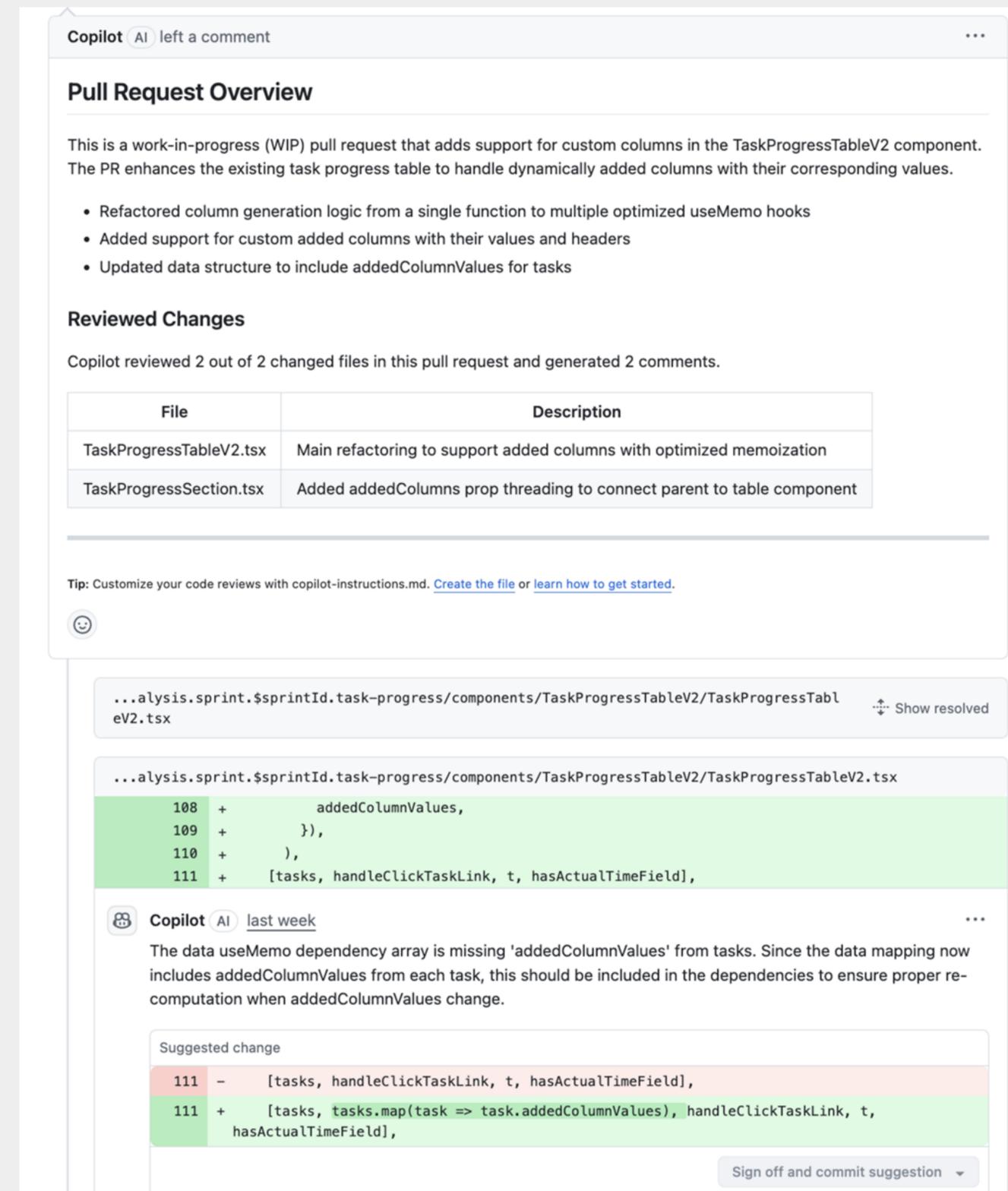
```
3 type UpdateCycleTimeGitHubExcludedDeploymentsParams =
4   UpdateCycleTimeGitHubGetDBReturn;
5
6 export type UpdateCycleTimeGitHubExcludedDeploymentsReturn =
7   UpdateCycleTimeGitHubGetDBReturn;
8
9 /**
10  * updateCycleTimeGitHubで使用するservice関数
11  * DBから取得したdataのうち、更新不要なデプロイを除外する (サイクルタイムが計測済みのもの)
12  *
13  * 除外対象
14  * - SourceChangeがないデプロイ (空デプロイ) →保持対象のCommit, PRが無いため
15  * - SourceChangeに紐づくサイクルタイム用PRがDBに存在するデプロイ →サイクルタイム用PRが計測済みのため
16  */
17 export const updateCycleTimeGitHubExcludedDeployments = (
18   params: UpdateCycleTimeGitHubExcludedDeploymentsParams,
19 ): UpdateCycleTimeGitHubExcludedDeploymentsReturn => {
20   - return;
21   - // 1. paramsのdeploymentsに対して、filter()
22   - // params.deployments.filter()
23   - //
24   - // 1-1. SC.length==0のデプロイはfalse
25   - // deployments.sourceChanges.length==0のデプロイはfalse
26   - //
27   - // 1-2. SCに紐づくサイクルタイム用PRが一つでもDBに存在するデプロイはfalse
28   - //
29   - // deployments.sourceChanges.cycleTimeGithubPullRequestAndCommits.pullRequestが存在するデプロイはfalse
30   - // 2. 返却
31
32   + const filteredDeployments = params.deployments.filter(({
33     sourceChanges }) => {
34     + if (isEmptyArray(sourceChanges)) {
35     +   return false;
36     + }
37     +
38     + const hasExistingCycleTimePR = sourceChanges.some(
39     +   ({ cycleTimeGithubPullRequestAndCommits }) =>
40     +     cycleTimeGithubPullRequestAndCommits.pullRequest,
41     + );
42     +
43     + return !hasExistingCycleTimePR;
44     + });
45   +
46   + return {
47     + ...params,
48     + deployments: filteredDeployments,
49   };
50 }
```

変更行数500行のPRに対し、人間が変更を加えたのは計21行のみ

GitHub Copilotで 軽微なコードレビューを定期的 的に実施

Navigator、Driver以外の人々が、実装中に
レビュー依頼を繰り返し実行し、
指摘内容をNavigator、Driverに共有

→レビュー修正単体の時間はゼロ



Copilot AI left a comment

Pull Request Overview

This is a work-in-progress (WIP) pull request that adds support for custom columns in the TaskProgressTableV2 component. The PR enhances the existing task progress table to handle dynamically added columns with their corresponding values.

- Refactored column generation logic from a single function to multiple optimized useMemo hooks
- Added support for custom added columns with their values and headers
- Updated data structure to include addedColumnValues for tasks

Reviewed Changes

Copilot reviewed 2 out of 2 changed files in this pull request and generated 2 comments.

File	Description
TaskProgressTableV2.tsx	Main refactoring to support added columns with optimized memoization
TaskProgressSection.tsx	Added addedColumns prop threading to connect parent to table component

Tip: Customize your code reviews with copilot-instructions.md. [Create the file](#) or [learn how to get started](#).

...alysis.sprint.\$sprintId.task-progress/components/TaskProgressTableV2/TaskProgressTableV2.tsx Show resolved

```
108 +     addedColumnValues,  
109 +     },  
110 +   ),  
111 +   [tasks, handleClickTaskLink, t, hasActualTimeField],
```

Copilot AI last week

The data useMemo dependency array is missing 'addedColumnValues' from tasks. Since the data mapping now includes addedColumnValues from each task, this should be included in the dependencies to ensure proper re-computation when addedColumnValues change.

Suggested change

```
111 -   [tasks, handleClickTaskLink, t, hasActualTimeField],  
111 +   [tasks, tasks.map(task => task.addedColumnValues), handleClickTaskLink, t,  
      hasActualTimeField],
```

Sign off and commit suggestion

操作の動きを再現してもらい、 テストシナリオを言語化して もらう



画面収録 2025-09-24 14.46.42.mov
ファイル

添付した動画は、agile effect上の操作キャプチャになります。
あなたはこの動画を解析し、操作に伴う、テストシナリオを作成して
ください。

⋮

ありがとうございます。これをスプレッドシートで履行管理できるテ
ストシートにできますか？

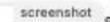
GPT Agentモード 出力結果

テストシナリオ：スプリント解析からDevOps解析への遷移と認証エラーの確認

目的

- 左側のAgile Effectメニューから「DevOps解析」を選択できることを確認する。
- DevOps解析画面で計測設定変更ボタンを押した際に、未認証状態の場合は401エラー画面が表示されることを確認する。
- エラー画面の「ホーム画面へ戻る」ボタンを押すと、スプリント解析画面に戻ることを確認する。

事前条件

- ブラウザで agile-effect.com にログインし、スプリント解析ページが表示されていること（左側には環境選択メニューやプラグインアイコンが並んでいる） 

手順と期待結果

手順	操作内容	期待結果
1	左側のプラグインアイコンバーから agile effectのコンパスアイコンをクリックする。	ロゴと「COMPASS」ドロップダウンが表示され、メニューに「スプリント解析」「DevOps解析」「設定」「ヘルプ」が並び 
2	メニューから「DevOps解析」をクリックする。	画面上部に「DevOps分析」と表示されたDevOps解析ページに遷移し、プロジェクト選択フィールド（例：cel_cms_front）、表示単位のドロップダウン（月など）、タブ「Four Keys/サイクルタイム」、4つの指標カード（デプロイ頻度・変更までのリードタイム・変

A	B	C	D	E	F	G	H
テストケースID	テスト名称	事前条件	テスト手順	期待結果	実施結果	ステータス	備考
3	De ア ツ	スプリント解析 ページが表示され ている	左側のコンパスアイ コンをクリックする	agile effectのメニューが表示さ れ、ドロップダウンによって 『スプリント解析』『DevOps 解析』『設定』『ヘルプ』が表 示される			
4	エ ボ	agile effectメ ニューが表示され ている	メニューから『 DevOps解析』をク リックする	DevOps解析ページが表示さ れ、『DevOps分析』というタ イトル、プロジェクト選択 フィールド、表示単位ドロップ ダウンが存在し、Four Keysタ ブが選択された状態でデプロイ 頻度・変更までのリードタイ ム・変更障害率・障害復旧時間 のカードとグラフが表示される			
TC002	DevOps解析ページ 遷移の確認	agile effectメ ニューが表示され ている	メニューから『 DevOps解析』をク リックする	DevOps解析ページ右 上の『計測設定を変 更』ボタンをクリック する	認証情報がない場合、401 Unauthorizedエラー画面が表示 され、『Unauthorized: token not found』のメッセージと 『ホーム画面へ戻る』ボタンが 表示される		
TC003	計測設定変更クリッ ク時のエラー確認	DevOps解析ページ が表示されている	DevOps解析ページ右 上の『計測設定を変 更』ボタンをクリック する	エラー画面の『ホ ム画面へ戻る』ボタ ンをクリックする	スプリント解析ページに戻り、 スプリントカードが表示される		
TC004	エラー画面からホー ム画面への遷移確認	401エラー画面が表 示されている	エラー画面の『ホ ム画面へ戻る』ボタ ンをクリックする	スプリント解析ページに戻り、 スプリントカードが表示される			

留意点

- スプリント解析画
- DevOps解析ペー
する。
- 401エラーが発生
認する。

リリースの速度を担保しながら 安全にデリバリーする



AeyeScan

AeyeScan（エーアイスキャン）とは、AIやRPAを活用したSaaS型Webアプリケーション脆弱性診断サービスです。Webセキュリティや開発などの専門知識が...

Aeye Scan 出力結果

 **LeveragesIncomingWebhooks** アプリ 10:16
AeyeScanからの自動通知

スキャン名
ID:527. Agile Effect CI/CD

メッセージ
スキャンが終了しました。

巡回開始時刻 / 終了時刻
2025/09/19 10:00 / 2025/09/19 10:02

スキャン開始時刻 / 終了時刻
2025/09/19 10:02 / 2025/09/19 10:16

スキャン結果

Critical: 0

High: 0

Medium: 0

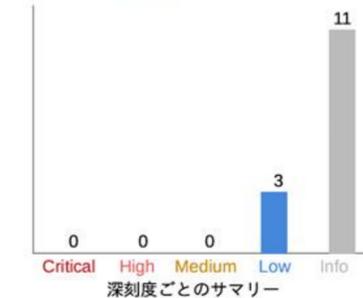
Low: 3

Info: 11

【Agile Effect】脆弱性のスキャン結果
以下URLから脆弱性診断結果を閲覧
<https://tool-AeyeScan-prd.s3.ap-nort>

スキャンサマリー

全体評価 **Low**



脆弱性の深刻度はCVSSv4 (<https://www.first.org/cvss/v4.0/specification-document>) に基づき以下の基準で設定しています。

深刻度	CVSSv4基本値	脆弱性に対して想定される脅威
Critical	9.0~10.0	・リモートからシステムを完全に制御されるような脅威 ・大部分の情報が漏えいするような脅威
High	7.0~8.9	・大部分の情報が改ざんされるような脅威
Medium	4.0~6.9	・一部の情報が漏えいするような脅威 ・一部の情報が改ざんされるような脅威 ・サービス停止に繋がるような脅威 ・その他、Critical/Highに該当するが再現性が低いもの
Low	0.1~3.9	・攻撃するために複雑な条件を必要とする脅威
Info	0	・その他、Mediumに該当するが再現性が低いもの

AX全体の効果：スピード（1/2）

リードタイムは、平均10.8日かかっていたものが、4.5日に。50%以下にすることに成功



AX前

AX後

効果



定量面

- ・コーディング時間は約60%増加（3.7h → 6h）
- ・レビュー待ち時間が約90%短縮（4.3h → 24min）
- ・デプロイ待ち時間が約60%短縮（10.3day → 3.8day）

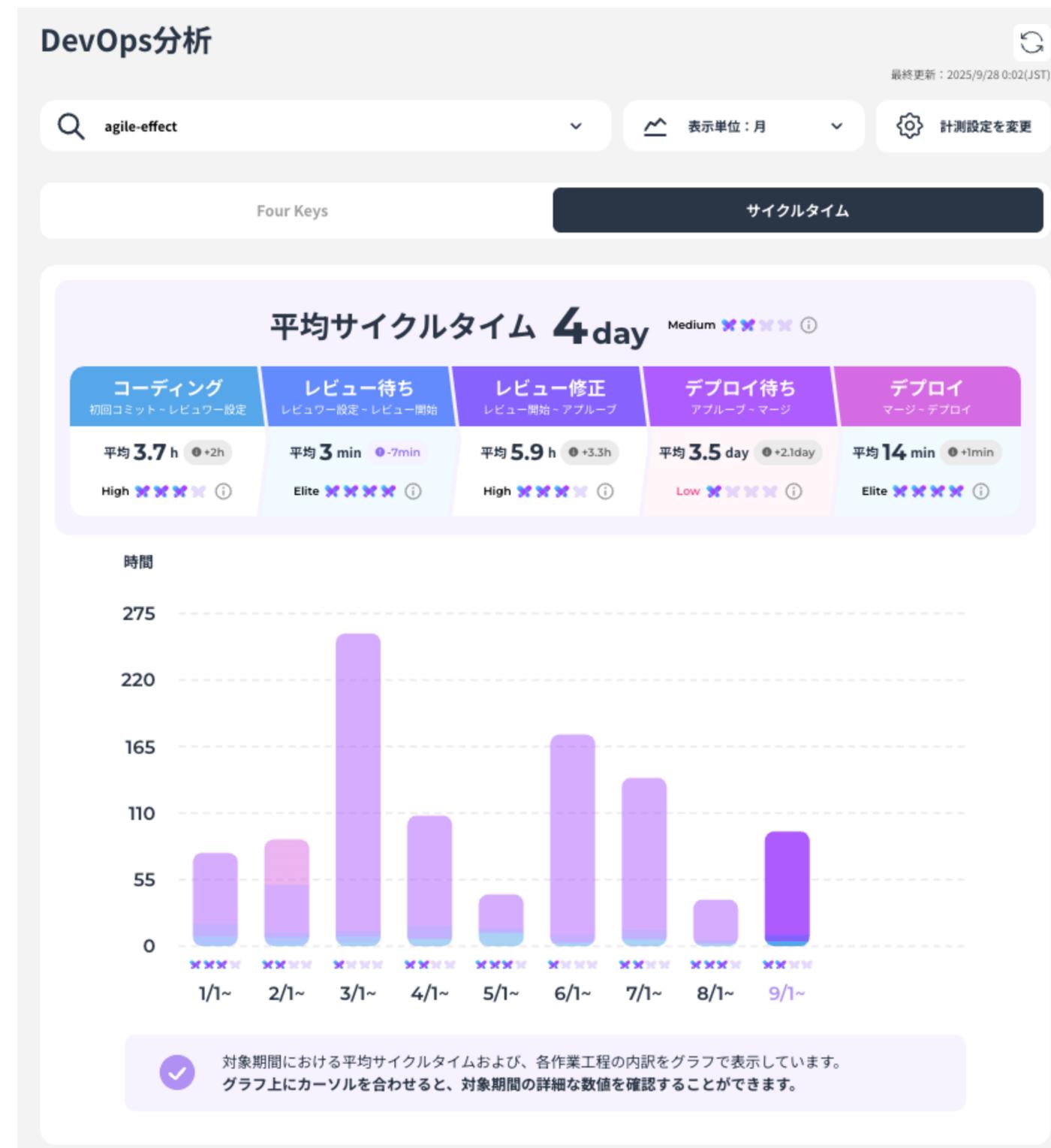
定性面

- ・属人性解消により、誰が何でもできる状態
- ・手戻りが大幅に減少
- ・主体性が自然と生まれる

リードタイムは遅くなると思っていたが、結果的に、早くなった。

AX全体の効果：スピード(2/2)

1つのリリースにかかる時間が4日程度に短縮し、ベロシティーは2.5倍に増加



AX全体の効果：品質

もともと重大な商用バグが0件をキープしていたが、AI導入後もそのレベルで維持



AX前

AX後

効果

変更障害率

障害の定義：デプロイ済みのプルリクエストにissueが起つた際、そのデプロイは「障害を起こした」と定義
算出方法：障害を引き起こしたデプロイ数 ÷ 合計デプロイ数
計上先：障害を引き起こしたデプロイの実施日に計上



定量面

- デプロイ待ち時間（実際はテストに費やした時間）が約60%短縮するも品質は維持できた（10.3day→3.8day）

定性面

- リグレッションテストなどは自動化によりストレスなく、新規機能開発に集中できる
- AIを過信せず、テスト観点の抜け漏れがないかを全員で確認する文化が出来上がったことで、全員のQAスキルが底上げされた

テスト項目数の増減はそこまでなかったが、AIの導入によって自動化が進み、E2Eテストの時間が短縮されたことで、同品質かつリリースまでの時間を大幅に削減できた。

自分のスケジュール管理をGPTに任せる

明日のやることリストです。

やることが多いので、それぞれ時間を見積もって、スケジュールを提案してください。

社内イベント周り

メールのテンプレート作成（次回イベント情報をくっつけたり）

ブース企画内容：ドット投票内容確定させる

11月イベントのQRコードを作る

リツイート用のサムネイル画像を作る

スポンサーにLinkedInで事前にご挨拶

チラシの発注遅れないように関係部署を牽制

イベントのやることリストをスプシにまとめる

トークスクリプトの整備（ペルソナごとのスクリプト）

17 明日のスケジュール提案

午前（集中系タスクを先に）

- 09:00–09:30（30分）
イベントのやることリストをスプシにまとめる
→ 全体を整理してから他タスクに入ることによって漏れ防止。
- 09:30–10:15（45分）
ブース企画内容：ドット投票内容を確定
→ 頭が冴えているうちに意思決定タスクを完了。
- 10:15–11:00（45分）
トークスクリプト整備（ペルソナごと）
→ 重めなので集中力が高い午前に割り当て。
- 11:00–11:30（30分）
メールのテンプレート作成（次回イベント情報を追加）
→ 具体が固まった状態で文章に落とし込む。

昼前後（軽めの作業タスク）

- 11:30–12:00（30分）
11月イベントのQRコード作成
→ 単発タスクで早めに完了可能。

（昼休憩 12:00–13:00）

午後（付外・調整系タスク）

自分のスケジュール管理をGPTに任せる

10/5の予定として、Google Calendarに組み込んで欲しい

📄 10/5（日）の予定をGoogleカレンダーにインポートできるCSVファイルを作成しました。

📄 [こちらからダウンロードできます](#)

Googleカレンダーに取り込む際は「設定 > インポート」からこのCSVを選択すれば、イベントとして追加できます。

← 設定

全般

カレンダーを追加

インポート/エクスポート

インポート

エクスポート

インポート

📄 パソコンからファイルを選択

カレンダーに追加
西慎一郎

iCalendar (ICS / VCS) または CSV 形式の予定情報をインポートできます。

インポート



Calendar interface showing a calendar view for October 2025 and a list of events for October 5th.

Calendar view: 2025年10月

月	火	水	木	金	土	日
29	30	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2
3	4	5	6	7	8	9

Event list for October 5th (GMT+09):

- 午前8時
- 午前9時: イベントのやることリ
- 午前10時: ブース企画内容：ドック 午前9:30～10:15
- 午前11時: トークスクリプト整備 午前10:15～11時
- 午後12時: メールのテンプレート
- 午後1時: 11月イベントのQRコード
- 午後2時: 昼休憩 午後12時～1時
- 午後3時: スポンサーにLinkedIn
- 午後4時: チラシ発注関連の調整 午後1:30～2:15
- 午後5時: リツイート用サムネイ 午後2:15～3時
- 午後6時: トークスクリプト整備 午後3時～4時
- 午後7時: 全体の確認&未完了タ

自分のスケジュール管理をGPTに任せる

あなたは私の秘書です。
いただいたスケジュール通りに進めていきたいので、それぞれ時間が来たらリマインドと次やることを教えてもらってもいいですか？
通知方法は、ChatGPT内でリマインドください。
リマインドは、終了時間が来る5分前にお願いします。

私から差し込みがあった際は、それも伝えるのでその場合は、スケジュールの調整案をください。

承知しました！

10/5（日）のスケジュールを秘書として ChatGPT内で終了5分前にリマインド します。

次の優先タスクに取り組む時間です：

1. 参加者の方に事前のアナウンス

30分以内の完了を目指してください。

自分のスケジュール管理 をGPTに任せる

おはようございます、今日は残りのタスクを開始します

おはようございます！☀️

昨日までに 主催イベントのタスクは完了、残りは デブサミ関連タスク ですね。

📅 今日やるべき残タスク（デブサミ関連）

8. 企画書に記載する
9. 大阪に共有する
10. チラシ発注が遅れないよう牽制
11. Connpass登録用QRコード+導線確認
12. DevSumi中の集客ツイート準備
13. リツイート用サムネイル作成

返信漏れがないかを確認

※GPTでコネクタ機能をONにする

Gmailで返信必要そうなもので、返せていないものない？

今後やること

SlackもMCP繋いだらできそう..

Gmailで返信必要そうなもので、返せていないものない？

Gmail と通信しました

受信トレイを確認しました。返信が必要そうなものを整理します 📌

📌 返信が必要なメール

1. 【ご対応依頼】 pmconf2025 スポンサー様の情報ヒアリング（期限 10/3）
 - 主催：プロダクトマネージャーカンファレンス実行委員会
 - 内容：ロゴ画像、リンク情報、ブース当日トークテーマなどをGoogle Drive上のシートに記入依頼
 - 締切が 10/3（金） までなので、急ぎ対応・返信が必要です
👉 これは最優先で進めた方がよさそうです。
2. 招待: pmconfのブースのトークテーマを3つ考える会（9/30）
 - 木村さんからのカレンダー招待
 - 参加可否を返答していないようです
👉 出欠の回答をしておく必要があります。

📌 返信不要・通知系

- 承諾: アーキテクチャカンファレンス申込締切 → 既に木村さんが承諾済み通知
- Your authentication code (OpenAI) → 認証コード通知
- 広告が承認されました (Meta for Business) → 通知のみ
- CodeZine, ラクシル, Peatix, Eight → プロモーション/情報通知で返信不要

👉 優先度としては、

1. pmconf2025 スポンサー情報の対応（期限10/3）
2. pmconfブーストークテーマ会の参加可否返信（9/30）

この2点に返答しておくのが良さそうです。

ご希望なら、**返信文案**も作成しましょうか？

課題に対するAIへの期待と導入効果

AIを導入することで品質を担保しつつ、各種スピードを速くすることが出来た



AIを導入する前

AIに期待していたこと

AI導入による効果

ベロシティー

モブワークを実施していたため、どうしてもベロシティー(1Sprintで完成させた仕事量)が少ない

モブワークをしてもコーディングの速度を上げ、ベロシティーを増やしたい

コーディング時間は、平均3時間/PRほど増えたが、ベロシティーは2.5倍に向上

リードタイム

リードタイムが長い(10.8日)

リードタイムを短縮したい

リードタイムは、10.8→4.5日以下に(レビューの待ち時間や、修正時間がなくなったため)

リリース頻度

週1~2回のリリースが限界で安定もしていなかった

リリース頻度を増やし、お客様に価値を多く提供したい

リリース頻度は、安定して週2回以上はできるようになった

品質

商用の重大なバグは0件
変更障害率15%以下

今まで通りの品質は維持したい
(変更障害率15%以下)

既存の品質基準を維持できた
(変更障害率15%以下)

agile effectは、

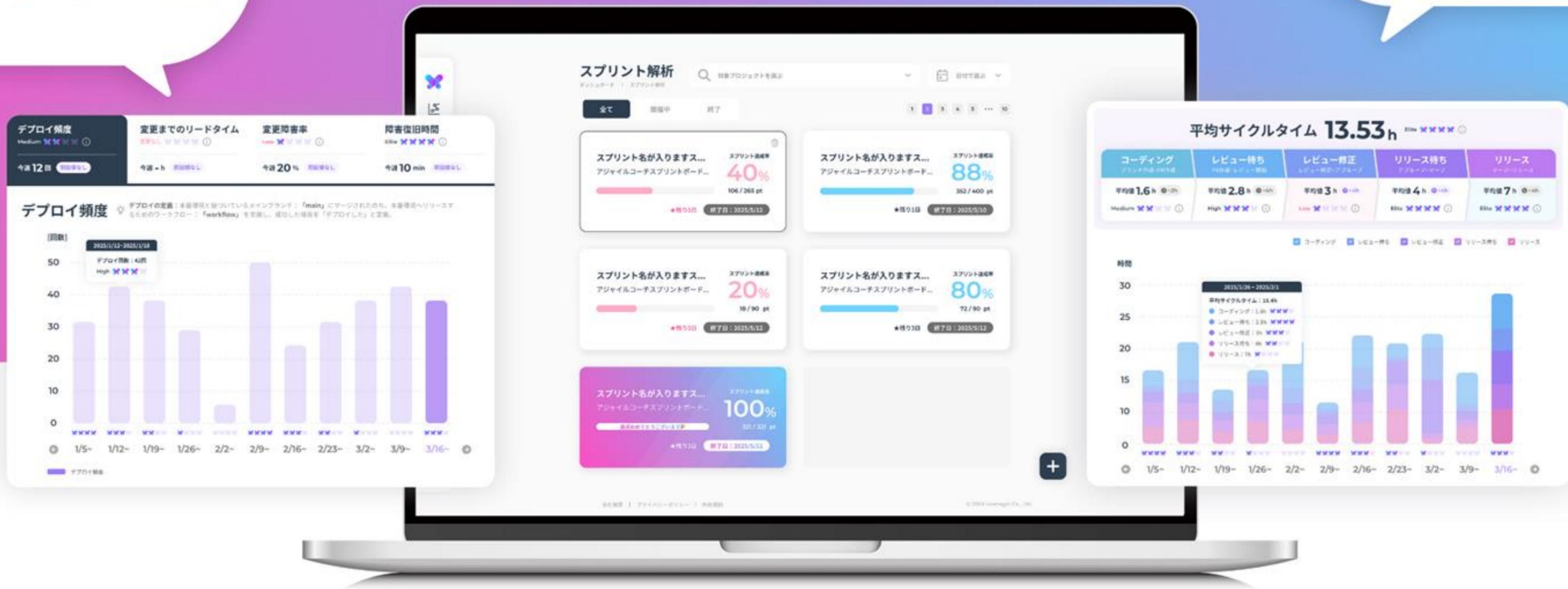
即効性と

導入の手軽さ

を感じるツール。

アジャイル開発、 お悩みのあなたへ。

スプリント期間を
短縮し、高速でPDCA
が回せるようになった！



開発プロセスを「見える化」し、意味のある改善を加速する **アジャイルエフェクト**



agileeffect
PRODUCED BY Leverages