組織で動かす AI-nize QA

AI を束ねる品質ハーネスの実装

株式会社メドレー ｜ 小島 大周 (@Daishu)

JaSST'26 東北 — 2026.05.29

自己紹介

小島 大周 (@Daishu) ｜ QA エンジニア ｜ AI x QA 推進

▼ 2022年 メドレー入社

- CLINICS (医療SaaS) にて、QA リード & AI x QA 横断
  - E2E テスト基盤と、QA の知見を集約した "qa-knowledge" を社内展開
  - 横断タスクの割合が増えていますが、インプロセス QA も大好き

▼ AI 利用歴

- Claude Code Max プラン 1 年ほど
  - 月間利用量で社内で 1 位になったこともあります (AI 好きです)

AI × QA で取り組んできたこと

qa-knowledge という AI x QA 基盤について

> プロダクト横断で QA エンジニアの知識と手順をまとめた Git リポジトリ

→ 2025 年 7 月頃に社内へ提供開始、運営して約 1 年経ちました

ブログもぜひ読んで頂けると嬉しいです！

qa-knowledge ができること ①

各プロダクトの知見や型を git リポジトリに集約 → プロダクト間で再利用・横展開

要素	内容	配布の仕組み
知識（ナレッジ）	テスト観点 / テストケース / ドメイン知識 / 学習パターン	md / yml
考え方	AI への判断基準・思考フレーム	Sessions / MCP
手順	QA 業務のワークフロー	Skills

> **各チームで磨いたベスプラを集約して、複数プロダクトの QA 活動を底上げする**

qa-knowledge ができること ②

テスト設計での気づき・学びを還元して、自己強化ループが回る

利用者は意識しなくても、AI の精度が勝手に上がっていく座組

今日お話しすること

開発工程に複数の AI を束ねる "品質ハーネス" を、組織でどう動かすか

	テーマ
①	品質ハーネスとは ― 工程ごとに複数 AI を束ね、責務を分離する
②	育てるループ ― 振り返りで観点・知見が循環し、ナレッジが育つ
③	QA の主戦場を広げる ― QA エンジニア = AI ハーネス
④	組織に届ける ― 作ったものを使ってもらう

> 品質ハーネス = AI をルール通りに動かす環境整備のこと

① 品質ハーネスとは

開発工程に複数の AI を束ねる

この章の問題意識

開発が AI で爆速になった。QA が追いつかない

- 各職種が AI を使って開発する
  - → 仕様も実装もレビュー依頼も、数分で降ってくる AI 時代
- 従来型の QA では捌けなくなる可能性
  - → AI 駆動開発において QA がボトルネックになる懸念

> 「AI を使って QA を効率化しよう」だけでは足りない
> → 各工程に AI を配置して、仕組みで回す必要がある

AI-nize QA = QA 全工程に AI を介在させる

品質ハーネスを実装する取り組みの名前です

> QA の全工程に AI を介在させる試みです (qa-knowledge を通して実践中)

なぜ「全工程」なのか

- 1 箇所に AI を入れても局所最適にしかならない → ワークフロー全体で設計する
- 各工程に AI がいることで、工程間の整合性も AI 同士でチェックできる

人間の介在度をどう決めるか

各工程に AI を配置する

説明責任が伴う判断は人、それ以外は AI に任せる

- 軽い判断（候補出し・差異検出・客観評価） → 🤖 AI が「考える」
- 重い判断（不可逆・説明責任・ドメイン文脈） → 👤 人間が「判断する」

QA の工程	AI がやること	人間がやること
仕様レビュー	仕様の抜け・曖昧さ検出	リスク判定・テスト方針
テスト設計	観点・ケース候補出し	採否・絞り込み
テスト実行	探索的テスト・差分検証	手触りの確認・結果の是非判断
振り返り	知見の抽出・分類	採用/却下の意思決定

「軽い／重い」の境界を定義することも、ハーネスの一部

自動テストで AI の出力を評価する

- 既存機能が壊れていないかは、自動テストに判断を任せられる
- 特に E2E テストは内部実装に依存しないので、実装の作り直しに強い
    - → AI がコードを書いては捨てる状況でも、E2E は壊れにくい

> - **E2E テスト基盤自体が、AI の出力を評価するハーネスになるので強化中**
> - **E2E に限らず、プロダクトのテスタビリティが高いことも重要**

どう AI を束ねるか

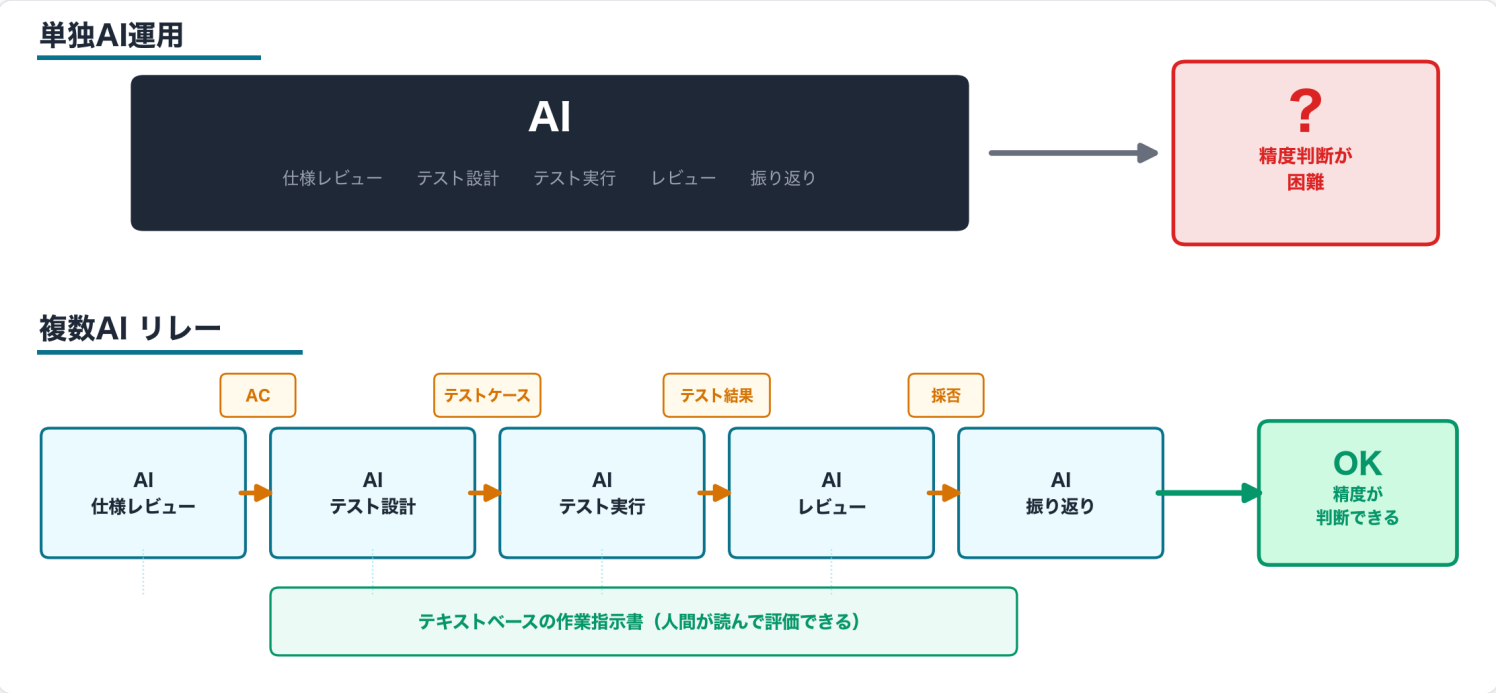
最初は 1 つの AI に全部投げていた

- テスト分析 → テスト設計 → レビューまで 1 チャットで AI と対話
  - チャットで出来上がったものを自分がチェック
  - AI の出力を AI がチェックしていない → 全て自分でレビュー

> タスクごとにセッションレベルで AI を分けることで、AI が AI を検証できるハズ

単独 AI 運用 vs 複数 AI リレー

タスクごとに AI セッションを分離して、リレー形式で引き継ぐことで精度を上げる



- AI 間でバトン (中間成果物・証跡) を渡すので、人間が評価しやすい足場となる
- セッションが異なるので、AI の成果物を別の AI で評価できる (LLM as a judge)

AI の成果物にどう向き合うか

AI が書いてくる量に、人間が追いつけない

- 各職種が AI をフル活用する時代
  - 要件定義もデザインも実装も、AI が書いている
  - 数分で何かがドサッと出てレビューに回ってくる
- 矛盾や事実誤認が混ざっていて、額面通りに受け取れない
  - 何より人が書いていないので「真剣に読もう」という気持ちが湧きづらい

> AI の成果物をどう受け止めるか・・

AI 成果物との向き合い方 ―― 3つの観点

観点	やっていること
バイアスを避ける	人間が先に見て、AI は後から漏れを拾う
成長を活かす	ジュニアは自分で作ってから AI に聞く
量に対応する	AI が生成して、人間は 1 つだけ読む

観点1：重要な判断は、AI を見る前に自分の判断を持つ

- AI は常にもっともらしいことを言う
  - 先に AI の評価を見ると「合ってそう」で流してしまう
- 重要な判断において、このバイアスは危険
  - → 先に自分で判断を持っておけば、AI の出力を「答え合わせ」に使える

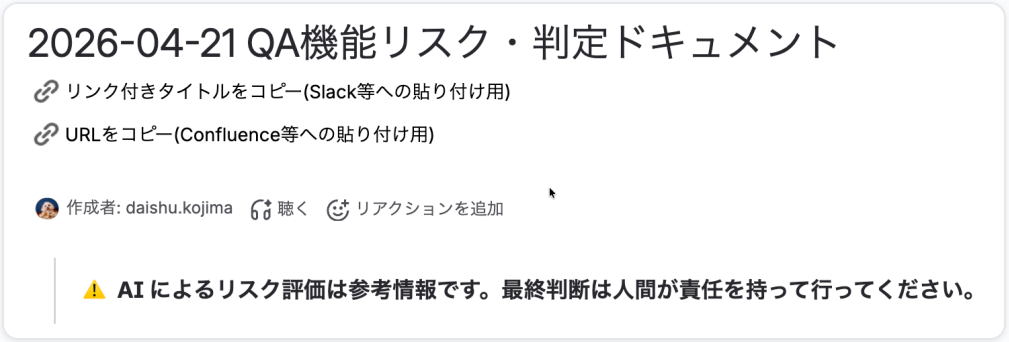
> 👤 人間が先に判断する → 🤖 AI は後から漏れを拾う役割

具体例：リリース前 QA レビューでの運用

リリース物の内容チェックでの具体例

1. リリースノートと PR を、人間と AI に同じインプットとして渡す
2. 👤 QA チームがまず人間だけでレビュー
3. 🤖 AI が同じ PR をリスク評価 → レポート出力
4. 人間のレビュー結果と AI のレポートを突き合わせて、漏れを拾う

qa-knowledge のスキル：リリース PR を AI がリスク評価 → テストの妥当性をレポート出力してくれる



観点2：スキル別の向き合い方

AI を先にするか後にするかは、スキルレベルと目的で変わる

ベテラン QA（スキル高い）

- 🤖 AI で叩きを生成（観点・ケース）
- 👤 人間が評価・絞り込み
- → 効率最大化

先述したケースは逆が望ましい

ジュニア QA（成長中）

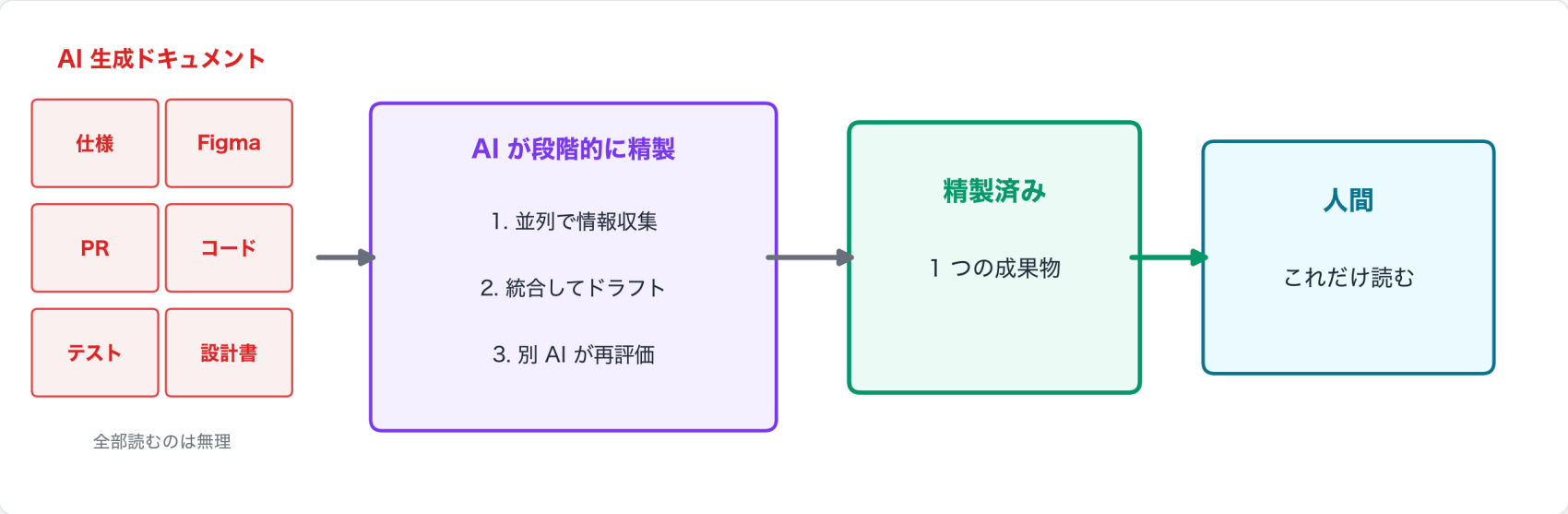
- 👤 人間がまず作る
- 🤖 AI が漏れチェック
- → QA エンジニアとして成長

AI は 24h いくらでも質問できる

- **私は先輩 QA の方々のレビューで成長したが、AI との関係も同じはず**
- **遠回りに見えるが、成長観点では重要 → 先輩 (AI) の考えを盗んでいく**

観点3：AI の生成 doc を 1 つにまとめて、物量に立ち向かう

AI によって生成されたドキュメントが各所にある状況：



- 複数の AI が並列で情報を集め、統合し、別の AI が再評価する
- 人間が読むのは、生成後の 1 つに絞る (or それを地図とする)

② 育てるループ

振り返りが観点を循環させる

この章の問題意識

ドキュメントは書いた瞬間から腐ってしまう

- ナレッジをまとめても、3ヶ月も経てば現状と乖離する
- 仕様のドキュメントが更新されない
- メンテを個人の努力に頼り続けるのは難しい → 仕組みで回したい

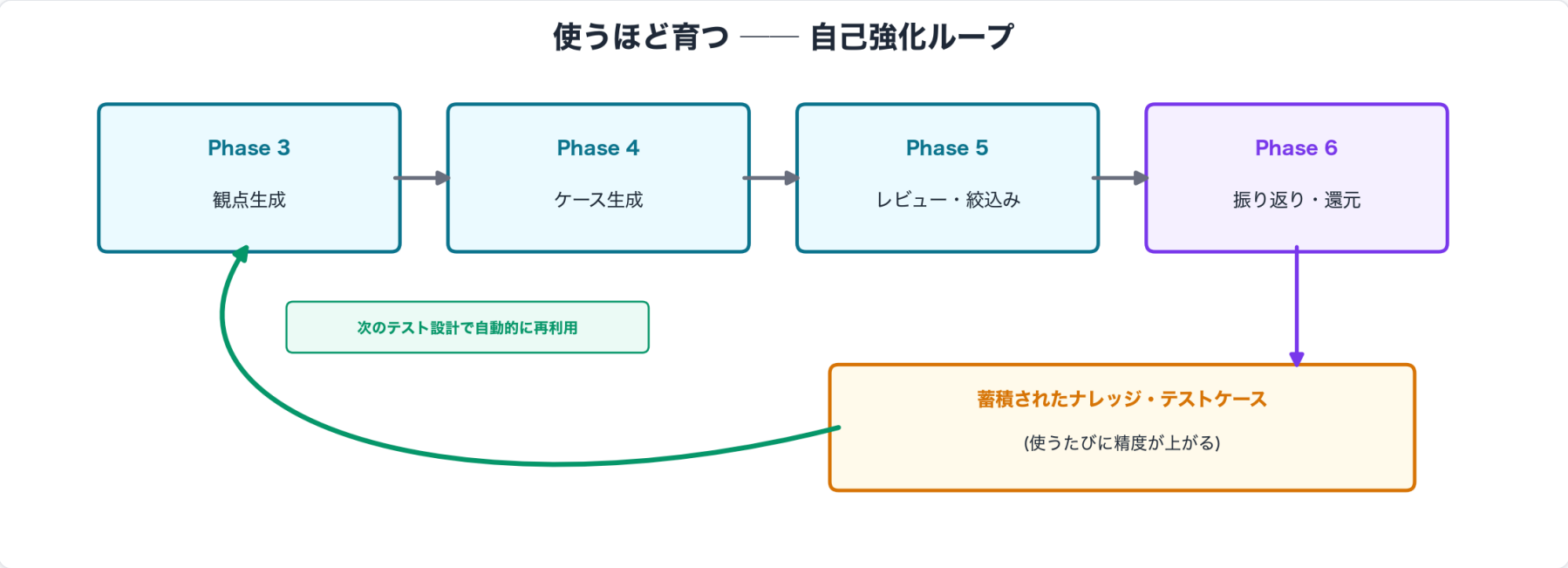
ナレッジは「使う」から「育てる」へ

	使うナレッジ（一般的）	育てるナレッジ（私の考え）
たとえると	辞書・マニュアル	畑
価値の最大	書いた時点	使われるたびに増える
時間経過で	古くなる = 腐る	自己強化ループで成長
「使う」と「育てる」	別行為	同じ行為

通常ドキュメントは書いた瞬間から腐っていくが、育てるナレッジは使うほど価値が増す

qa-knowledge の自己強化ループの実装

振り返りで観点・知見が循環する



- テスト設計などでナレッジを使いながら、qa-knowledge に蓄積
- 振り返りが「使ったら捨てる」を「使ったら育つ」に変える

ナレッジを「正式化」する月次 Retro

qa-knowledge では AI がファシリを行い、ナレッジ候補を昇格させるフローがある

※ 対象: hotfix / bugfix で蓄積された知見も含む

ステップ	担当
候補ナレッジ (テスト中に AI が気づいたパターンの蓄積) や障害内容を 1 件ずつ提示	🤖 AI（ファシリ）
文脈・出典・使われた回数、障害の再発防止としての価値などを説明	🤖 AI
採用 / 却下 / 修正の判断	👤 参加者
合意したものだけ `knowledge/` として昇格してコミット	🤖 AI
使われなくなったナレッジを廃棄	🤖 AI

> **AI が淡々と事実ベースにファシリ進行するので、健全なポストモーテムになる (感情抜き)**

月次 Retro のイメージ

【グループ 1/4】🔴 緊急対応 PR（2件）

障害から学ぶナレッジは価値が高いため、最優先で確認します。

2. Product A #HOTFIX-02: [HOTFIX] グループに異なる種別のアイテムが追加できてしまう
   📌 推奨: プロダクト固有 → product-notes.md
   💡 理由: プロダクト固有のグループ仕様に関する観点
   📊 スコア: 92 / 100

   📝 学びの内容:
   【事象】　　新画面でグループに異なる種別のアイテムが追加できてしまう
   【なぜ漏れた】リアーキ時の新旧機能差分テストが不足
   【今後の観点】
   - リアーキ時は旧画面との機能差分テストを実施（新旧で同操作 → 結果比較）
   - グループ種別 × アイテム種別のクロステスト
   - タブ横断のフィルタリング確認: 主要タブだけでなく関連タブからの検索も必ずテスト
   - 特殊ケース網羅: 例外フラグ付きアイテム、特殊バリアント、外部区分
   - 通しテスト: グループ作成 → メイン処理 → 完了処理でエラーが出ないことを確認

課題：AI はプロダクトを知らない

- テスト観点のナレッジは育ったが、成果物にズレがある場合あり
- AI はプロダクトを「どう操作するか」を知らない
- 「このボタンを押す」「この画面から遷移する」という実画面の知識がない
  - コードから 0 ベースで把握するのは高コスト → 類推となりやすい
  - テスト手順がズレる、探索的テストが浅くなる

> 「何をテストするか」が揃ったら、次は「どう動くプロダクトか」を AI に渡す

E2E SpecBook について

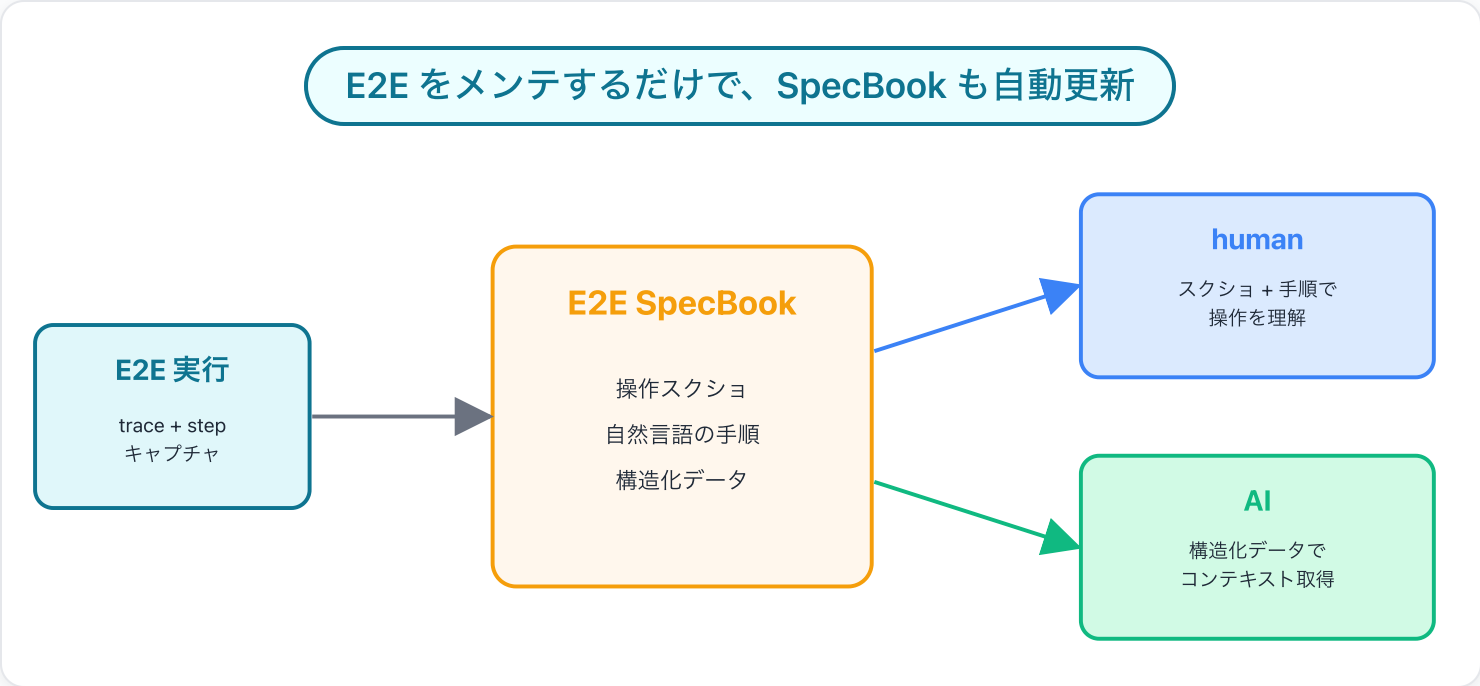
qa-knowledge と連携する、E2E テストから自動生成される画面マニュアル (自作内製ツール)

> **ドキュメントは腐るが、E2E は腐らせない。なら E2E テストを軸にすればいい**

- E2E SpecBook とは
  - E2E 実行時のキャプチャとステップから、画面マニュアルを自動生成
  - 日々、E2E をメンテするだけで、マニュアルも勝手に最新になる
- E2E は毎日実行されるので、常にプロダクトの変更に追従する
  - → ドキュメントが常に最新 = living documentation

E2E から画面マニュアルを自動生成する

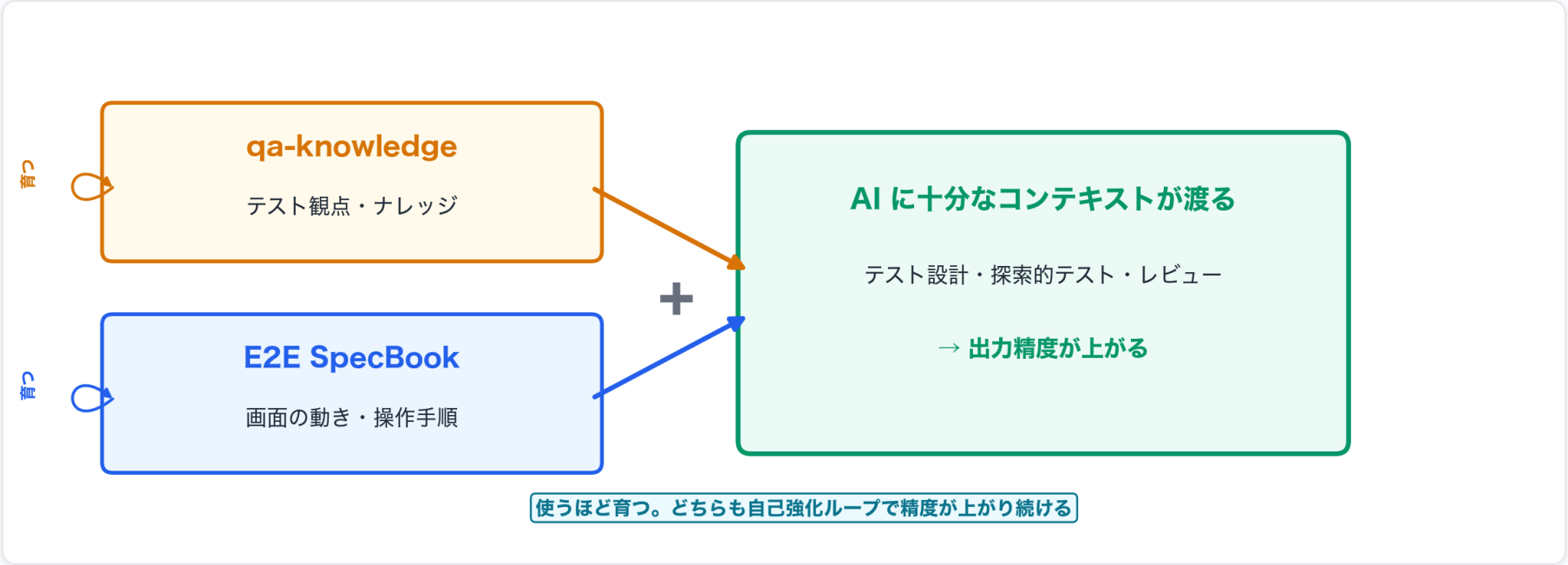
AI 用のコンテキストだけでなく、人間のオンボーディング資料としても使える



🔥 技術詳細は、別途テックブログで共有予定です

ナレッジ + 画面仕様 で AI の精度が上がる

ナレッジ = 見るもの / 画面仕様 = 動かし方、両方を渡す



- コンテキストが揃うと、AI がプロダクトの中身を掴んで動く
- どちらも自己強化ループで育つので、メンテはほぼ乗せておくだけ

③ QA の主戦場を広げる

QA エンジニア = AI ハーネス

QA エンジニアはもとからハーネスをやってきた人

ハーネス = AI をルール通りに動かす環境整備のこと

ハーネスとなる対象が「人・プロセス」だけでなく「AI」にも広がっただけ

機能	従来：対人ハーネス	AI時代：AI ハーネス
集める	工程・観点をストックする	複数の AI エージェントを配置する
方向付ける	テスト方針・リスク判定	受入基準・ルール定義
守る	品質ゲート・リリース判定	ガードレール・トレーサビリティ
回す	振り返り	知見の蓄積・還元ループ

> **QA がやってきたことは、そのまま AI ハーネスになる**

プロジェクト QA を続けながら、主戦場を広げる

「QA担当」から「品質ハーネスを設計・運用する人」へ

私のこれまでの主戦場

- インプロセス QA
- 個別案件のテスト設計・実行
- 1 案件 = 自分 1 人の手で品質を見る

並行して広げた主戦場

- 工程に AI を束ねるプロセス設計
- ナレッジが育つループの運用
- 1 工夫を組織の仕組みに変える

- **これからもインプロセス QA は継続したい**
- **現場にいるからこそ、必要なハーネスが見えると思っています**

AI 時代のシフトレフト

- AI の出力精度は、インプットの質で決まる
- 前工程で品質を作り込めば、後工程の AI 生成物の精度が全て良くなる
- QA が前段に関わること自体が、AI ハーネスの精度を上げるはず

> **シフトレフトを極めてインプットの質を上げることも、主戦場としていきたいです**

④ 組織に届ける

ハーネスの難所 — いきなり使ってもらえるものではない

> **ハーネスは AI を束ねるより、人間に乗ってもらうことのほうが難しい**

最初の数ヶ月は、自分しか qa-knowledge を使っていなかった

- プロダクトごとにチーム構成・文化が違う → 1つのやり方を押し付けられない
- テスト設計が日常タスクではない職種には、そもそも刺さらない
- git 前提のリポジトリ提供 → 「難しそう」で離脱される

現在地 — 日常運用に組み込まれている段階

どう浸透させたか

- Skill・MCP 提供
  - → テスト設計に限らず、貯めたナレッジを使える
- 社外で発信
  - → 社外の反応で、社内での認知も拡大
- QA エンジニア向けにハンズオンを実施
  - → QA 所属チームのプロダクトへ浸透

チーム構成	効果
QA が居るチーム	qa-knowledge を活用して品質保証を加速 (AI 駆動開発のリズムに乗る)
QA が居ないチーム	一定品質で QA タスクが回せる

qa-knowledge は「プロダクト」として作ってきた

社内ツールでも「使ってもらう」ためにはプロダクト思考が要る

観点	内容
ユーザ視点	ツールを使うと、テストケースの生成精度が上がる
使ってもらう工夫	複数の入口 (Skill / MCP / CLI) を用意して、誰でも入りやすい
独立性	開発リポと切り離す → どのプロダクトでも使える
育つ仕組み	自己強化ループを組み込む → 使うほど賢くなる

> インプロセス QA で培った、良いユーザ体験を作る感覚が土台になっている

初めて品質ハーネスを作る方へ

token 溶かしながら学んだこと 💸💸

段階	やること
1. 任せる	まずなんでも AI に任せてみる
2. 知る	AI の特性を体で覚える（忘れる / 非決定性 / 文脈依存）、AI に質問しながら学ぶ
3. 対応する	外部記憶（md 等）、手順を作る、レビューポイントを決める
4. 育てる	気づきをルールに溜める → 自己強化ループ
5. 配る	ルールを組織に配れる形にする

> **任せる → 知る → 対応する → 育てる → 配る = 最小単位の組織ハーネス**

基盤がなくても始められる。まず 1 つの工程で AI に任せてみるところから

(小話) AI x QA 推進というロール

「組織全体で使えるツールを作ろう」と、AI x QA の基盤開発を進めた
→ 「AI × QA 推進」という肩書きも自分で名乗り始めた
→ 当初は AI 駆動開発のスピード感に対する危機感から始めたもの

始めてみると、頭の中で描いていた理想の QA プロセスを AI で形にできる
この面白さがモチベーションになって、今も取り組んでいます。

> **QA における AI 活用は各社で実践中と思うので、事例交換できたら嬉しいです🤷**

ご清聴ありがとうございました