



AIに任せた品質は、誰が見立てるのか

AI時代のテストマネジメント

JaSST'26 Niigata | 中野 直樹 (株式会社LayerX)

自己紹介



中野 直樹

バクラク事業部 QA部 部長

略歴

エンタープライズシステムの開発・PMを経てQAエンジニアへ転向。ネット証券でのテスト自動化・テストリード、不動産テック企業でのQAマネージャーを経て、2023年4月LayerX入社。事業成長を支えるQA組織の構築を主導。現在はプロダクト全体の品質戦略・テスト戦略の策定に加え、新規プロダクトのQA支援も担当。

社外活動

- NPO法人ソフトウェアテスト技術振興協会（ASTER）理事
- ソフトウェアテスト技術者資格認定（JSTQB）技術委員
- ソフトウェアテストシンポジウム JaSST Tokyo 実行委員
- 『ソフトウェアテスト教科書 JSTQB Foundation 第4版』共著



Mission

**すべての経済活動を、
デジタル化する。**

人類の未来をより良くする。

そのために私たちは、テクノロジーの可能性を探求し、
経済活動における複雑で大きな課題に挑む。

仕事や暮らしの中にある摩擦が解消され、
それぞれの創造力が発揮されている。

そんな希望あふれる優しいデジタル社会を、
未来に残していくために。

「すべての経済活動を、デジタル化する。」をミッションに、複合的な事業を通して日本の社会課題を解決し、AIの力で人々の創造力がより発揮される未来をつくります。

バクラク事業



バックオフィス向け
AIエージェントサービス

Ai Workforce事業



エンタープライズ向け
AIプラットフォーム

Fintech事業



資産運用サービス
ALTERNA（オルタナ）

セキュリティ事業



AIエージェントによる
セキュリティ検証サービス

AIエージェントで 業務を完全自動化する

バクラクは、バックオフィス業務を自動化する
AIエージェントを提供するクラウドサービスです。

AIが日々仕事を進め、人は結果を受け取り判断に集中する。
作業を楽しむのではなく、その作業が不要になる状態を。
月次を待つ経営から、日々意思決定できる経営を実現します。

中小企業から大企業まで、20,000社を超えるお客様の
働きやすい環境づくりと事業成長を支援しています。



Agenda

目次

- **01 AIに任せる前に**
- 02 全部をAIに任せてみたら
- 03 テスト工程ごとのポイントを考える
- 04 任せた後に、見えてくるもの
- 05 まとめ

AIはテストの作業をこなせる。しかし、品質に責任は持てない

- AIがテストプロセスの各工程の作業をこなせることは、確認できている
- しかし、当然、品質に責任を持つことは、AIにはできない
- AIに作業を任せ、人が品質に責任を持つための技術が、テストマネジメント

AIを試すのは簡単、仕組みにするのは難しい

生成AIを業務で
試している

89%

生成AIが
全社に広がった

15%

AIの信頼性を
課題に挙げる

60%

AIエージェント案件が
中止される、と予測

40%超

出典: World Quality Report 2025-26 (Capgemini / Sogeti / OpenText, 2025-11) 品質エンジニアリング領域の調査。上級役員2,000人超、22カ国、10業種
Gartner press release, 2025-06-25。企業のエージェント型AIプロジェクト全般が対象

今日「超有能な新人（AI）」があなたのチームに来たら

できること

- 仕様書を渡せば、その日にテストケースが出る
- 24時間動く。何人分でも並行して働く
- 疲れない、嫌がらない、文句を言わない

できないこと

- 会社やプロダクトの事情を、
言われなくても分かる
- 朝会やオフィスの会話から、文脈を拾う

新人に作業を任せるとき、私たちが無意識にやっていること

新人に任せるとき	テストマネジメントでは
何を任せて、何を任せないか決める	① テスト計画に、何を書くか
仕様書に書いていない、ユーザーの事情を伝える	② テスト分析・設計に、何を入力するか
何をもって作業完了とするかを決める	③ テスト実行で、何を成果とするか
作業結果を説明してもらう	④ テストレポートで、何を報告させるか
成果物を確認して問題を指摘する	⑤ 成果物を、どうレビューするか
終了後に振り返ってプロセスを改善する	⑥ テストの精度改善で、何を測り、次にどう戻すか

人にやさしい任せ方は、AIにもやさしい

- ただし「言わなくても伝わる」ことはない
- 人の新人は朝会に出て、オフィスの会話で文脈を拾ってナレッジを蓄積するが、AIはできない
- 任せ方は同じでも、伝え方は変える。AI協業前提の設計を考える必要がある

Agenda

目次

- 01 AIに任せる前に
- **02 全部をAIに任せてみたら**
- 03 テスト工程ごとのポイントを考える
- 04 任せた後に、見えてくるもの
- 05 まとめ

「全部をAIに任せたら」という思考実験

1 制約を外す

人の工数という制約を、一度外す

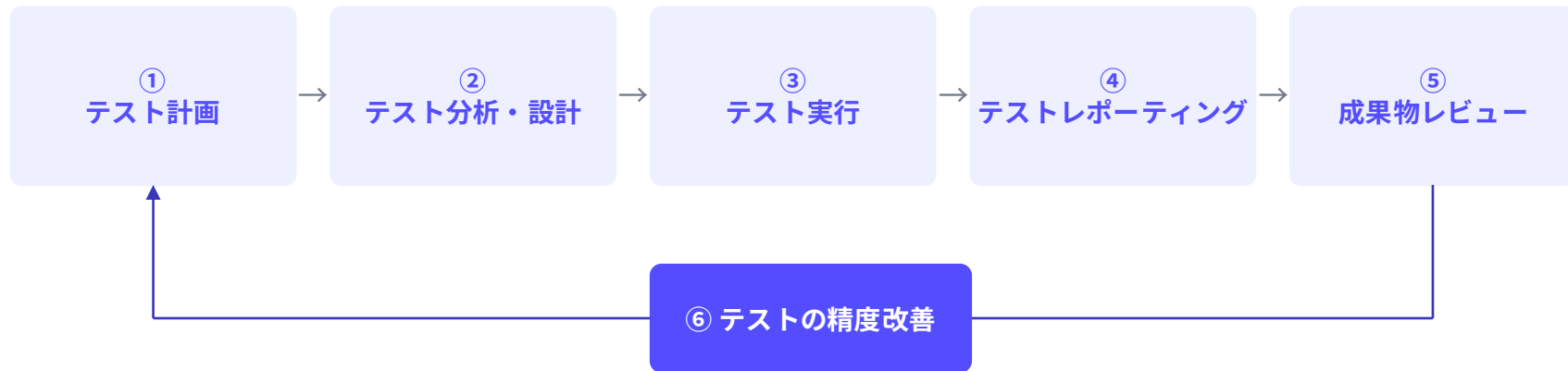
2 テスト工程で考える

各テスト工程で「制約が外れたら、どう設計し直すか」

3 振り返る

全部をAIに任せた後に、何が見えてくるか

AI前提のテストプロセス



⑥ テストの精度改善：各工程のAIの判断（リスクの見積もり、テストの十分性、判定、報告）の精度を測り、次の計画に戻す

従来のテストプロセスで、現場が最適化してきたこと

	現場でやってきたこと	かけずに済んだ工数
リスク	リスクを発生確率×影響度の点数で評価していた。本来は本番障害が起きたときの損失額で見積もれる	測定して、更新し続ける工数
文書	テスト計画は要点と参照リンクだけにして、詳細は他の文書とコミュニケーションでカバー	書いて、保守する工数
判断	チームで顔を合わせているから、判断基準、優先順位、許容範囲はコミュニケーションで決め、全部は書かずに済んだ	明文化して、合意する工数

人の工数という制約が、外れた

	Q 品質	C コスト (人の工数)	D 納期	S スcope
これまで	調整	固定	調整	調整（諦める側へ）
いま	調整	制約ではなくなった	調整	調整（広げる側へ）

Q=品質（ここでは外部品質） C=コスト（人の工数） D=納期 S=スコープ（どこまで確かめるか）

人の工数の制約が外れ、取れるテスト戦略も広がった

- APIテストで厚く、E2Eは薄くする配分（テストピラミッドの形）は、自動テストの技術としては以前から可能だった
- その戦略を取りづらかったのは、E2Eを減らしてもAPIテストで守れると、確かめる余力が人にはなかったから
- つまり「新しい戦略が生まれた」のではなく、人の工数の制約が外れたことで、テストピラミッドの理想の形に早く近づくようになった

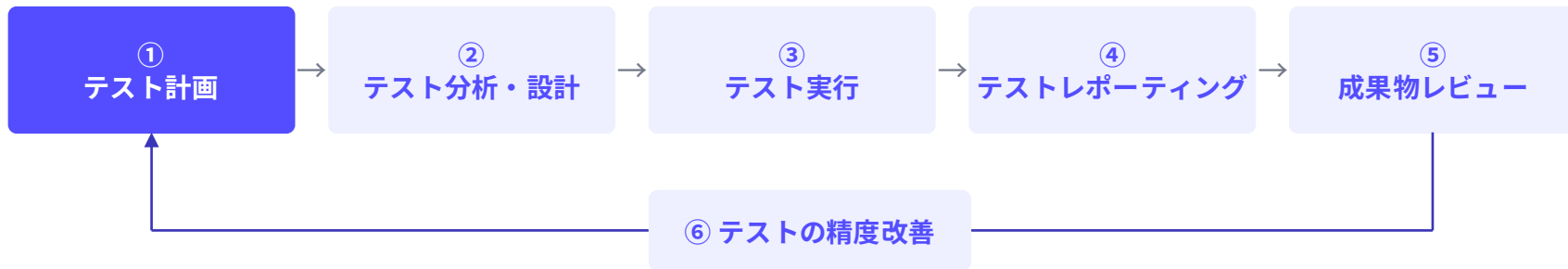
Agenda

目次

- 01 AIに任せる前に
- 02 全部をAIに任せてみたら
- **03 テスト工程ごとのポイントを考える**
- 04 任せた後に、見えてくるもの
- 05 まとめ

ポイント① テスト計画

- 同じ規格の2つのテスト計画。AIと協業するならどちらか
- アジャイル型の計画が薄くて済むのは、なぜか
- テスト計画として、AIには何を渡すべきか



同じ規格に載っている、2つのテスト計画

A: Agile Corporation

約1.5ページ。要点だけ書き、詳細は他に分散

テスト計画: 対象
文脈・テストアイテム・スコープ
チームとコミュニケーション
ステークホルダー
リスク→「ストーリーカード参照」
テスト戦略（箇条書き）
・見積り／TDD／回帰
・リスクに応じた技法
・Done基準・日報報告

テスト戦略（つづき）
・イテレーション末の要約
・開始/終了基準はDoDに内包
「その他は組織的テスト戦略とチーム憲章のとおり」

B: Traditional Ltd

約7ページ。項目を網羅して一つに集約

改訂履歴
承認欄
参考文献
目次

概要
文脈：
レベル/タイプ
テストアイテム
テストスコープ
テストベース

前提と制約
ステークホルダー
コミュニケーション

リスク管理
プロダクトリスク
×10
(確率×影響=点数)
軽減策

プロジェクト
リスク×7
テスト戦略：
レベル/タイプ
成果物/技法

開始基準・終了基準
独立性/メトリクス
データ/環境
再テスト/回帰
中断・再開/逸脱

活動と見積り
要員：役割
採用/教育
スケジュール

規格に載っている架空の2組織のサンプル（ISO/IEC/IEEE 29119-3:2021 Annex E）の構成と分量を図にしたもの。規格の原文ではない

AIと協業するなら、どちらのテスト計画がいいか

A

Agile Corporation 型

約1.5ページ

B

Traditional Ltd 型

約7ページ

Agile型（A）は薄いのではなく「分散」している

- Agile型の情報は、ユーザーストーリー、チケット、wiki、組織的テスト戦略に分散している
- 書かれていない前提は、口頭で補われている
- スプリントプランニングや朝会に出られないAIへの、その前提の渡し方を検討する必要がある
- チームの暗黙の判断基準やルールも、明示する必要がある

AIが事前に知っておくべきこと

戦略のコンテキスト

- テストスコープ
（品質特性を含む）
- プロダクトリスク分析の
結果と根拠
- テストレベルの厚み配分と、
その理由

システムのコンテキスト

- アーキテクチャと、テスト容
易性
- 変更頻度が高い場所、
壊れやすい場所
- 過去の障害と、再発防止

判断のコンテキスト

- ユーザーにとって、
どうあるべきか
- 何を許容し、
何を許容しないか
- 迷ったときの優先順位。
誰に、いつ上げるか

Traditional型（B）の7ページでも、抜けやすいもの

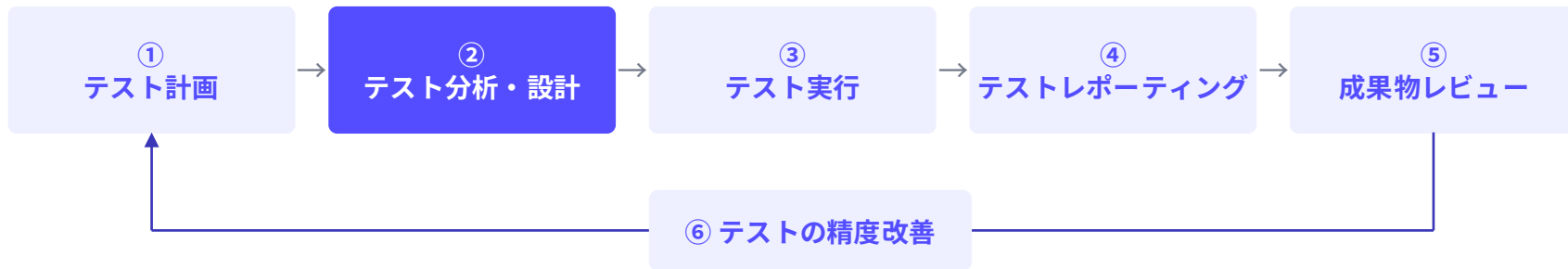
- リスクの点数の根拠（なぜその点数なのか）
- テストレベルの厚み配分が、組織的テスト戦略とずれている理由
- 壊れやすい場所と、過去の障害からの学び
- QAの視点で考えた「ユーザーにとってどうあるべきか」
- 不具合の判定材料や品質の許容範囲といった、暗黙のルール

読み手も書き手もAIである前提の新しいテスト計画

- Agile型（A）もTraditional型（B）も、そのままではAIにとって十分な入力にならない
- Agile型の分散した情報と、Traditional型の網羅的な記載に加えて、プロジェクトの仕様書、プロダクトコード、ユーザー視点（QAエンジニアの知見）などを渡していく必要がある
- また、暗黙的な方針の文書化や過去のテスト資産などもナレッジとして活用できる可能性がある

ポイント② テスト分析・設計

- テスト分析・設計は、全部AIに任せられるか
- AIに何を渡せば、設計を任せられるか
- AIと人は、それぞれ何を担うか



テスト設計を、2つに割ってみる

決める

何を、どこまで確かめるかを定める
リスクと判断基準が要る

AIに任せますか？

書く

決めたことを、テストケースに書き出す
仕様があれば書ける

AIに任せますか？

AIが担う／人が担うの分かれ目は、工程の中にもある

- テスト設計のような一つの工程の中にも、AIが担えるタスクと人が担うタスクがある
- テスト設計に限らず、テスト計画、テスト実行などでも同じ

AIがテスト設計をするための入力として必要なもの

- 従来のテストベース（仕様書など）
- プロダクトリスク
- 判断基準（リスクレベルに応じた網羅条件、テストの優先順位付けの方針など）
- プロダクトコード（人の工数の制約が外れ、AIにコードを読ませて抜け漏れを補完できるようになった）

AIが担うこと、人が担うこと

AIが担う

- 開発タスクのチケットとプロダクトコードから仕様を調べ、対象機能を洗い出す
- テスト計画とテストケースを作る
- ブラウザで実行し、記録を残し、一次判定して報告する

人が担う

- プロダクトリスクと判断基準を渡す
- AIが作った計画を確かめ、実行に進めてよいか決める
- 判定に迷った項目を、最後に判断する
- 探索的テストを続ける

ポイント③ テスト実行

- テスト実行を任せたら、何をもって「できた」とするか
- 任せる前に、何を決めておくか
- AIの「完了」の報告を、そのまま信じてよいか



AIに任せるテストの「実行」と「検証」を分けて考える

- 成果を「実行した記録 (🟢)」にすると、「終わった」以上の情報がなく、AIが何をしなかったかが見えない
- 環境起因のスキップと、判断によるスキップの区別もつかない
- 成果を「検証の記録」にすると、何を確かめ、何を正常と定義したかが残るので、AIに「何を担保したか」を報告させられる

実行はAIに任せ、何を評価し、正常とするかは人が決める

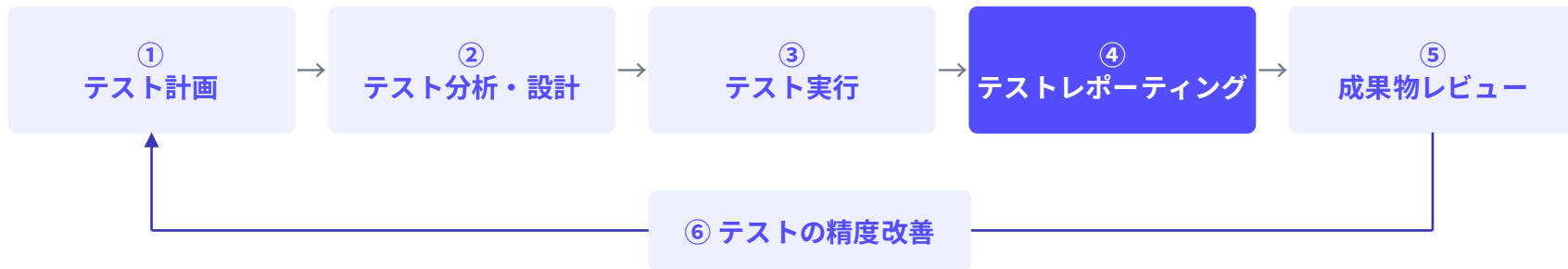
- 何を検証し、何を正常とし、異常をどう知らせるかをAIに任せる前に決める
- 人が見ているうちは設計の甘さを人が補ってしまう
AIに任せて人が見なくなって初めて、評価の設計の差が結果に現れる
- AIは心配性で、小さな気になる点まで全部報告してくることがあるので、報告を受ける人とのコミュニケーションの設計も重要

実例：テストの完了ログは、実施の証拠ではない

- ・ 夜間のAIリグレッションテストで、ブラウザに接続できず、全ケースがBLOCKで止まった
- ・ 次の実行はその結果を「実施済み」と読み、「テスト完了」と報告した
- ・ 人が詳細なテスト結果を確認するまで、気づけなかった

ポイント④ テストレポーティング

- AIのレポートから、人は何を知りたいのか
- そのために、AIに何を報告させるか



報告させるのは「残存リスク」と「判断の根拠」

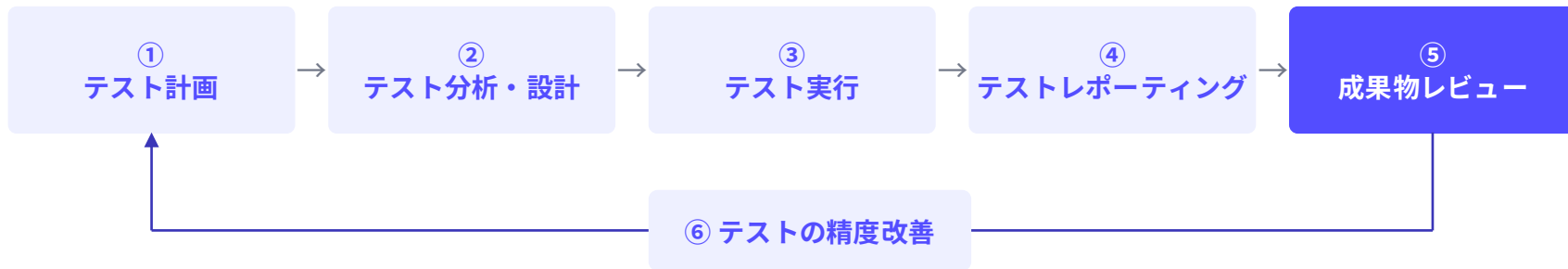
- 「何件通ったか」は実行の記録。責任を持つ側が知りたいのは「何が残っているか」
- 残存リスクの形で報告するには、リスクからテスト結果までのトレーサビリティが必要
- AIに、「やらなかったこと」「減らしたこと」と根拠をレポートに含めさせる

AIに出させるテストレポートの中身

報告させること	中身
確かめたこと	どのリスクを、どのテストで、どう判定したか
確かめていないこと	未実施、BLOCK、スキップと、その理由
減らしたこと	削除や降格をしたテストと、その根拠
人の判断が要ること	判定に迷った項目、許容範囲の境目
残存リスク	リリースに向けて、何が残っているか

ポイント⑤ 成果物レビュー

- AIが作った計画やテストケースを、人は全部読むのか
- 人はAIの何を確かめるのか。そのために、AIに何を残させるか



中間成果物は、AIの判断を人が確かめるための根拠になる

- 最終成果物だけでは、AIの判断の妥当性は確かめられない
- 「なぜそのテストを選んだか、削ったか、バグを取りこぼしたか」を
さかのぼれないと、改善が回らない
- 各工程の成果物の質が、次の工程の入力の質を決める

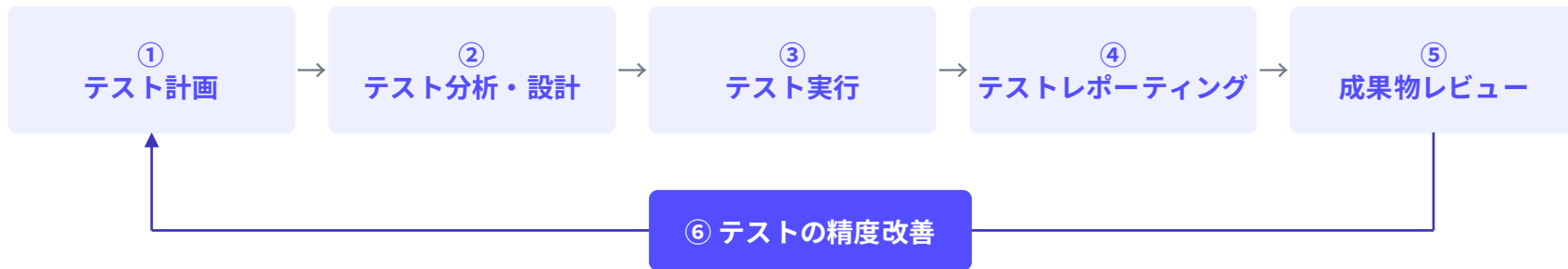
どの成果物も、人が読む分とAIが使う分を分けて設計し直す

- 人は全部を読まない。人が読む成果物は、判断に要る情報に絞る
- AIが使う成果物は、精度を出しやすい形で残し、トークン効率も見る
- フォーマット、置き場所、ツールの選定に加え、
成果物を1箇所にまとめられない場合はAPIやMCPなどの通信要件も考慮が必要

ポイント⑥ テストの精度改善

ここでいう精度：各工程のAIの判断（リスクの見積もり、テストの十分性、判定、報告）と、実際の結果とのずれの小ささ

- テストを減らす判断は、誰がするか
- テストが十分かどうかを、何で測るか
- 測った結果を、どう次の計画に戻すか



テストの十分性が最も問われるのは、テストを減らす判断

- テストケースを増やすときは、誤っても増えるだけなので、AIに案を出させて、後から人が間引けばよい
- テストケースを減らすときは、誤ると検出能力が落ちる。
落ちたことに気づきにくいので、人が根拠を残して決める
- リスクスコアを下げる、テストレベルを落とすなどの判断も、人が根拠を残して決める

十分性を測る「真に有効なカバレッジ」は、まだ見つかっていない

- コードカバレッジは、ユーザー価値で測る時代には効かなくなった
- 機能、ユースケース、リスク。
代わりの指標は数えられても粒度と重みづけが測る人でぶれる
- まだ答えはない。誰が測ってもぶれない「次のカバレッジ」の発明が、QAに求められている

テストの精度改善は、AIが提案し、人が判断することが望ましい

AIが担う

- 不具合や本番障害、実行ログを集め、AIの判断のずれを分析できる資料にまとめる
- リグレッションテストや自動テストの追加案を出す
- 上流で防げた不具合を洗い出し、テストプロセスをシフトレフトする改善案を出す

人が担う

- AIが見逃した不具合や本番障害が、AIの判断のずれか、渡した入力不足かを見立てる
- AIの判断（リスクの見積もり、十分性、判定、報告）のどこを直すかを決める
- 直した判断基準が効いたかを、次の結果で評価する

AIの判断の良し悪しを、AI自身に評価させない。
直した判断基準は言語化して、skills（AIへの指示書）などの仕組みに組み込み、
次の①テスト計画に反映する

Agenda

目次

- 01 AIに任せる前に
- 02 全部をAIに任せてみたら
- 03 テスト工程ごとのポイントを考える
- **04 任せた後に、見えてくるもの**
- 05 まとめ

6つの工程のポイントに、共通して見えてくること

- どの工程でも、評価する側がボトルネックになる
- AIに任せても、品質は勝手には上がらない
- 最後の判断と責任は人に残る

どの工程でも、評価する側がボトルネックになる

- 一を言えば十をやる人が、できる人だった。いまは、AIから千が返ってくる
- 人が判断しきれない量の出力がAIから返ってきた瞬間、ボトルネックは評価する側に移る
- だから、人が評価の仕組みを設計することが重要

AIに任せても、品質は勝手には上がらない

品質に効かない任せ方の例

- 人の作業プロセスを、そのままAIに移植する
- 人が評価できないことを、AIならできるはずだと任せる
- 口頭で伝えてきた判断基準や文脈を、書き出さずにAIに任せる

最後の判断と責任は人に残る

- 責任とは、判断を誤ったときに損害を引き受け、ステークホルダーに説明し、次の判断を変える権限を持つこと
- AIは三つのどれもできない。完全自動運転でも、事故の責任は車には帰属しない
- だから、AIがどれだけ賢くなっても、最後の判断と責任は人に残る

「全部任せる」を進めた先にあったのは、全自動ではなかった

- 入力と出力が直接つながる作業は、AIに任せられる未来が見えている
- 一方、テストのモニタリングとコントロールのように、
同じ入力でも状況で動きが変わる仕事は、任せる難易度が高い
- モニタリングとコントロールなど複雑な判断と対応ができるエージェントは、まだない
- 全自動を前提に進めた結果、全自動はまだ遠く、設計と判断の大変さが分かった

43年前に書かれた、自動化の皮肉

- 設計者がオペレーターの作業を無くそうとしても、
自動化の方法を思いつけなかった仕事はオペレーターに残る
- 身体的な技能は、使われないと衰える。
それでも、異常時には経験者を呼ぶか、自ら引き継ぐことが期待される
- 仕事の易しい部分を取り去ることで、
自動化はオペレーターの仕事の難しい部分を、より難しくしうる

原文 (Lisanne Bainbridge, "Ironies of Automation", Automatica, 1983) 日本語訳は発表者

"the designer who tries to eliminate the operator still leaves the operator to do the tasks which the designer cannot think how to automate."

"physical skills deteriorate when they are not used" / "...they may be expected to call a more experienced operator or to take over themselves."

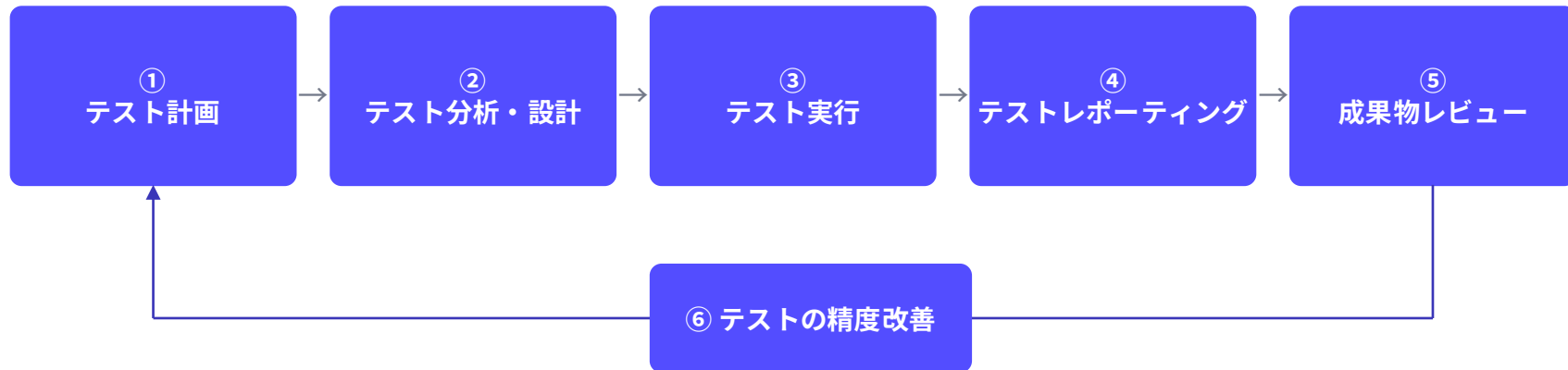
"By taking away the easy parts of his task, automation can make the difficult parts of the operator's task more difficult."

Agenda

目次

- 01 AIに任せる前に
- 02 全部をAIに任せてみたら
- 03 テスト工程ごとのポイントを考える
- 04 任せた後に、見えてくるもの
- **05 まとめ**

6つの工程を、見てきた



AIに任せるときの判断は、テストマネジメントの知識で答えられる

工程	ポイント	対応するシラバスの章
①計画	AIには何を渡すべきか	1.1.1 テスト計画の活動／1.4 プロジェクトテスト戦略
②分析・設計	AIに何を渡し、人は何を担うか	1.3 リスクベースドテスト（1.3.2 品質リスクの識別、1.3.4 リスク軽減）
③実行	何をもって「できた」とするか	1.1.2 テストモニタリングとコントロールの活動／1.6 テストツール
④レポーティング	AIに何を報告させるか	2.1.3 テスト報告／1.1.3 テスト完了の活動
⑤成果物レビュー	AIの何を確かめ、何を残させるか	2.3.5 欠陥レポート情報／2.3.6 プロセス改善アクションの定義
⑥テストの精度改善	十分性を何で測り、次にどう戻すか	2.1 テストメトリクス／1.5 テストプロセス改善（1.5.4 ふりかえり）

参照: JSTQB Advanced Level シラバス テストマネジメント Version 3.0 (J04、2026-06)

AIに任せられる範囲は、QAエンジニアの知識とスキルで決まる

- テスト戦略、リスク、カバレッジ、品質基準などのテストマネジメントの知識とスキル
- テスト分析、テスト設計の手法を使うスキル
- ユーザー理解（プロダクトがユーザーにとってどうあるべきかの解像度）
- 暗黙的な判断基準や口頭の文脈を、AIが読める形に言語化するスキル
- 自動テスト、CI/CD、AIを含む各ツールの知識と、AIへの指示や評価を設計するスキル
- エンジニアリングの知識（アーキテクチャ、テスト容易性、ログや監視）

明日、チームで実践するために

- 1 自組織のテスト計画は、誰に向けて書かれているか
- 2 これまで口頭で伝えてきた文脈のうち、文書として書き出すべきものは何か
- 3 AIの報告と実態のずれを、誰が、何と突き合わせて検出するか
- 4 テストの十分性を、どのように判断するか
- 5 テストを減らす判断を、根拠を残して決められる準備があるか
- 6 改善した判断基準を、skillsなどに組み込む仕組みがあるか

AIはテストの作業をこなせる しかし、品質に責任を持つことはできない

品質に責任を持つための技術は、テストマネジメントとして既に体系化されている。
学べば、誰でも手に入る

AIで激動の時代だからこそ、このカオスを楽しんでいきましょう