



AI時代におけるテストの基礎の 再定義

株式会社MIXI 開発本部 たんぽぽ室 QAグループ SDETチーム/QAEチーム マネージャー
松谷峰生



我々が
実現したいことは何も変わっていない。
変わるのは実現手段である。

自己紹介

株式会社MIXI 開発本部 たんぽぽ室 QAグループ
SDETチーム / QAEチームマネージャー
松谷 峰生 (まつや みねお)

趣味：
ロボット開発、小説執筆、マンガ執筆
DIY、プラモ、ゲーム開発

ペット：
フクロモモンガ

僕が描いている商業マンガが出ていたりします。
今連載マンガ2本を持っています。

- あなたの知らないテストの言葉
- マンガではじめるMagicPod入門

マンガではじめる MagicPod入門

ソフトウェアテストを初歩から学べる人気マンガ「テスターちゃん」の作者Matsuさん特別描きおろし！
テスターちゃんの仲間たちと一緒にMagicPodの使い方を楽しく学びましょう。

あなたの知らない
テストの言葉
連載

第8回
AIレッドチーミング
AI Red-Teaming
松谷峰生氏

マンガでわかる ソフトウェア
テスト入門

テスターちゃん Vol.2

松谷峰生氏



ソフトウェアテストのことがマンガで
わかる入門書の第2弾が登場！
社員教育の補助教材としても使える！
ソフトウェアテストに関するコラムも充実！



おっさんQAトークというYouTubeをはじめました

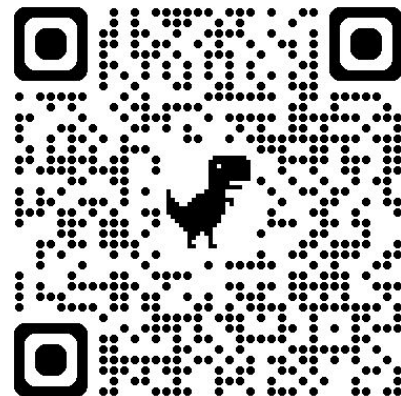
僕とさかたさんというQAの方で「おっさんQAトーク」というYouTubeをはじめました！

10分くらい、テーマについておっさん2人が好き勝手に話すというものです(笑)
聞き流し用です。

基本的には「AI時代の～」文脈の話をする予定です。

第1回「AI時代のQAのキャリア」

第2回「AI時代テスト実行者はどう
すればいいの？」



松谷とAI ~AIまわりでよく声をかけられる人です~

2017~2018
スマートスピーカーのQAマネージャー

言い方で出力が変わるなど、今もみんなが苦戦している部分と当時から向き合う

2018~2023
QA4AI設立メンバー

AIプロダクト品質保証ガイドラインの Voice User Interface領域を執筆

2018~2019
AIに対する品質保証についての講義/講演

IT検証フォーラムや宮崎大学で AIに対する品質保証で講演や講義

2022~2023 AIスタートアップにjoin

gpt-2を触っていた

2022
JaSST'22 ShikokuでAIについての招待講演

AIプロダクトの品質保証について

2024~2026 AIロボットの品質保証

メインはハードウェアだったけど



講演の全体像

AIへの向き合い方

QA・テストの仕事と役割

AI時代に必要なテストの基礎



AIは何を奪うのか？ ～AIの現状を把握しよう～

QA/テストの仕事がAIに奪われるという話を聞くようになった



とあるテストエンジニア x月x日

AIがテスト観点を山ほど出してくれるようになった！ もうAIでいいのでは



とある自動化エンジニア x月x日

自動テストのコードをAIが秒で作ってくれた！
私はコーヒー飲みながら見てるだけ



とあるQAエンジニア x月x日

全然自分よりイイカンジのアドバイスするよな…。AIに聞くでよくない？



とあるテストエンジニア x月x日

もう普通のテスト実行もAIがやってくれる時代になりそう…



文章の作成

コーディング

長文の要約

デザイン

アイデア出し

ファイル操作

お絵描き/動画生成

ブラウザやPC操作

etc...



単純作業

これまで
自動化を目指した領域



知的作業

これまで
我々が侵害されないと
思っていた領域

テスト成果物
生成

テスト観点出し、テストケース生成、テストデータ生成など。
たたき台作成が速くなっている。

自動テスト
補助

E2E自動テスト生成、自己修復、ビジュアルリグレッション。
リグレッションテスト作成と保守の重さを下げている。

判断補助

失敗原因分析、flaky検知、品質ダッシュボード作成。
大量の結果から見るものを絞り、意思決定しやすくする。

テスト計画

あまり事例を見ない。品質目標は人が決めて欲しい。
リスク分析はAIが良い仕事をするように見えている。テストのアプローチは、人は自分の領域依存に寄りやすいが、AIは各テストレベルやテストタイプを幅広く出せる。PJの都合を細かく与える必要はある。

テスト観点出し

周りでもよく見るAI活用。一般的な観点を幅広く出すことができる。
これまで起きた気を付けるべきissueやよくあるCSお問合せ、暗黙の仕様などのドメイン観点はコンテキストとして与える、人が補完する必要がある。
モデルによってはごっそり観点が漏れることがあるので要注意。

テスト設計

テストケース生成と同時に行われていそうなパターンはよく見る。
境界値や代表値は問題なく出せる。デシジョンテーブルテスト的なパターンも出せているように見えている。組み合わせが必要な部分は出せているか、ペアワイズで減らしたほうがいいのか、状態遷移図や表でのパターンの部分は確認したほうがいい。
gpt-5.5ではパターンがごっそりない場合を確認している。

テスト実装

コードでのテストの生成はAIの得意分野。人用のテストケースもテスト設計が出来ていれば出せると思うが、手順は与えるコンテキストによるため、実際に実行できる手順なのかは確認したいところ。(松谷は手順不要な確認でばかり作ることが多いため未確認)

テスト実行

E2E自動テストの実装・実行はよく目にする。

人用のテストケース実行のAIエージェントも、松谷が開発したものなどがあるため後ほど紹介。

また探索的テスト実行するAIエージェントも弊社では実働している。

テストレポート

情報まとめはAIが得意とする領域のひとつである。

普通に生成させてしまうと情報を全部盛り込もうとするため、skillsやテンプレートを使って行うほうが良い。

手動用のテストケースを実行できる AIエージェントを開発しました。
スポンサーセッションで5分で簡単にお伝え予定です。

実際のテスト102件を対象に、ツールが想定した通りに動くかの実験を行いました。

検証結果 (gpt-5.5)	件数	割合
意図した通りのテストができた	87 / 102	85.2%
意図した通りのテストができなかった	15 / 102	14.8%

ツールが意図した通り正しくテスト実行できたか。
バグ検出精度ではない

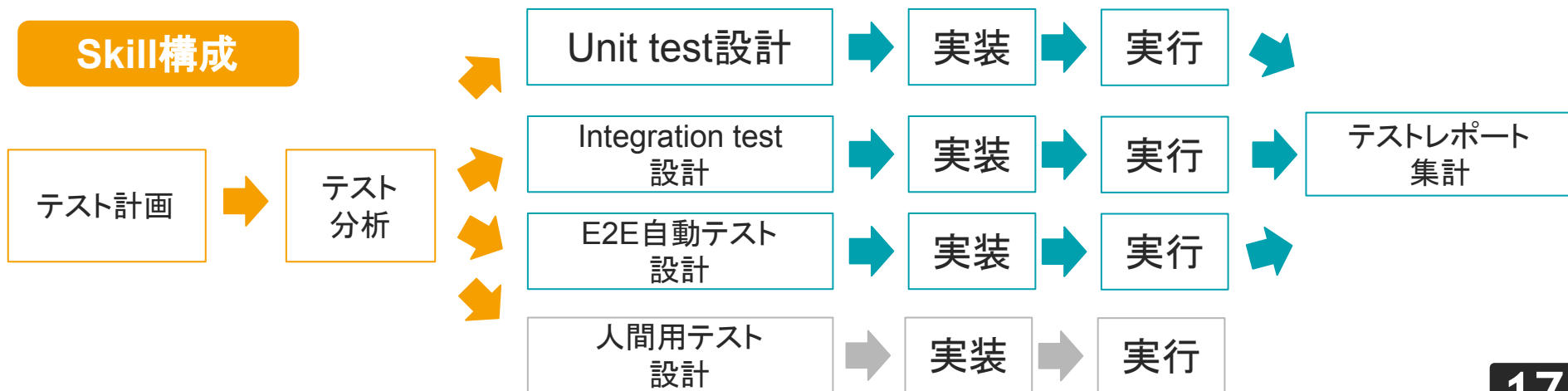
計画からレポートまでを全自動で実行するSkill

各Skillのオーケストレーションにより**計画から分析・設計・実装・実行・レポートまでを全自動実行**するSkillを開発しました。

Unit／Integration／E2E／人間実行を分け、**中間成果物を作り**次工程に渡しながら進みます。

テストベース/リスク→観点→設計→項目と**追えるように紐づけている**ことが特徴です。

※Appendixに説明を入れています



**AIは我々が行ってきた各作業を
できるようになってきました。**

**それによって我々の仕事が
奪われてしまうのでしょうか？**



AI時代の 考え方の再定義

AIは道具です

AIを過度に擬人化してしまうと、人と同等のことができると思っています。

結果、まるで自分より優れていると思っている人が同じ部署に来て自分の仕事に手を出してきた時のように、過度に不安に思ってしまったります。（または過度な期待）

**道具は使うものです
道具は使う人次第です**

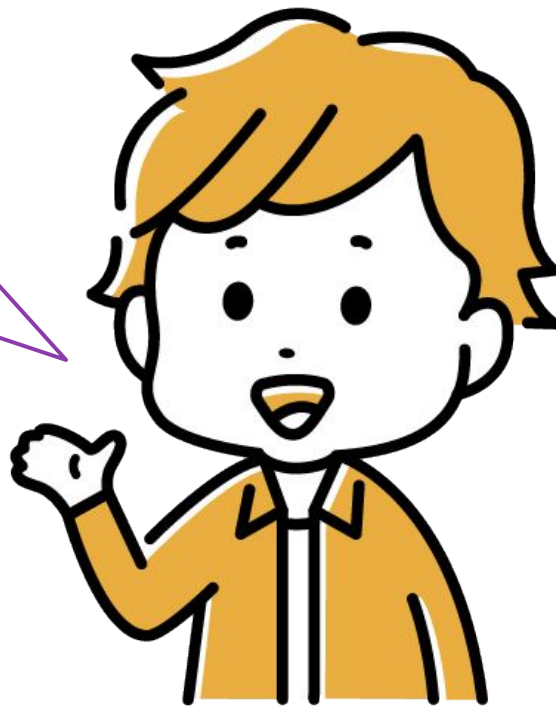
AIはとても便利です。便利なのですが、やはり道具なのです。
その道具をどのように使うかは、人次第なのです。

もう少しエンジニアっぽく言うと、

**AIという道具を、何に対して、どのように適用するか
を設計するのは人の仕事**

「AIに全部任せちゃう」という設計をするぞ！

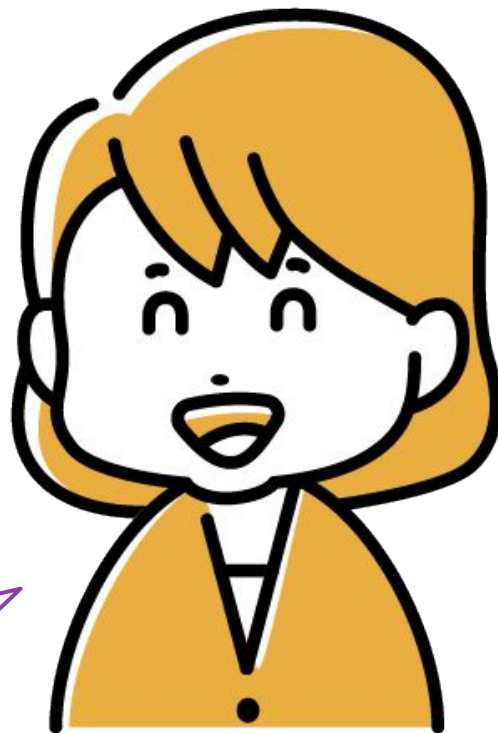
僕は細かいことを考えるのが好き
じゃないので、
**AIに全部任せるとい設計をしま
すね！**



AIで動くハサミを作ったぞ！
もう美容院もいらないし、全部この
ハサミでいいんじゃない？

ハサミよハサミよハサミさん。
髪が伸びてきたし、ちょっと切って
よ。

AIはすごいから全部お任せ！



これで坊主刈りになったとしても「ハサミが悪い！」と言ったところでどうにもなりませんよね。

AIは道具であり責任能力はありません。どのように使うかは人次第であり、どうなったかもその道具を使った人の責任 です。

**AIという道具を使って得た結果も責任も
使うと決定した人にある**

AIは道具であり、動作や結果の責任は判断した人にある



どこはAIでどこは人がやるといった、戦略は人の仕事

AIをどのように適用するかといった、プロセス設計は人の仕事

AIの権限やガードレールといった、環境設計は人の仕事

※これらを全てAIに考えてもらえますよね？は、AIに全て考えてもらうというプロセス自体を考えたのは人

人が起点。人が道具を使い目的をなす。

**AIは道具である以上、起点は人。
目的を果たすため、人が道具を上手に使う。**

結局、人が何かの目的を達成するために、人が適切な道具を選び、人が適切に道具を使うのです。

原始時代。あなたは**料理担当で、中でも火起こしに時間をかけていました。**木の棒をグルグルして火を起こすのですが、コツも必要で専門職なのです。ですがある時、なぜか突然「ライター」が出てきました。**誰でも簡単に火が使えるようになりました。**

あなたは「仕事がなくなってしまった…」と意気消沈です。

周りを見ると、**みんな思い思いにマンモスの肉をライターの火で直炙り** して「誰でも料理できるやん！」とやっています。

あなたは「それでいいんだっけ？」と感じつつも、火が簡単に起こせてお肉を誰でも焼けるので、自分の仕事は終わった...と思いました。

あなたの仕事の本当の価値は？

料理担当のあなたの価値は、木の棒を回して火を起こせることではなかったはず
です。

料理担当の本当の価値は、

**火をどう使えば、お肉が美味しく、安全に、みんなに満足される形にな
るか知っていること**

なのです。

**奪われるのは仕事そのものではない
仕事だと思い込んでいた一部の作業である**

**AI時代に生き残る人は
火起こしにこだわる人ではなく
料理人として再定義できる人**

あなたの「火起こし」に当たる部分はなんですか？
あなたの「料理人」にあたる仕事の本質は何ですか？



QA/テストエンジニアの 活動領域の再定義

品質活動 = テスト実行ではない

予防的・プロセス指向

品質マネジメント活動

検出・評価・是正的

品質保証(QA)

プロセス改善

標準化

欠陥予防

etc...

品質コントロール(QC)

テスト

形式的手法

シミュレーション

プロトタイピング

ル(QC)

テスト

コードでのテスト

コンポーネントテスト

コンポーネント
統合テスト

システム
テスト

システム全体での
テスト

外部システムもつなげ
てのテスト

システム
統合テスト

受け入れテスト

受け入れ当事者が
判断するためのテスト

形式的手法などetc...

システム テスト

テスト計画

テスト分析

テスト設計

テスト実装

テスト実行

テスト完了

コンポーネントテスト

コンポーネント
統合テスト

システム
統合テスト

受け入れテスト

品質マネジメント活動

品質保証(QA)

プロセス改善

標準化

欠陥予防

etc...

品質コントロール(QC)

テスト

コンポーネントテスト

コンポーネント
統合テスト

システム
統合テスト

受け入れテスト

システム テスト

テスト計画

テスト分析

テスト設計

テスト実装

テスト実行

テスト完了

ココ！

形式的手法などetc...

もしあなたが
「私の仕事はテスト実行専門だし」
と思っているなら視野を広げてみてください。

今あなたが見てる世界は本当のごく一部で
広大な未踏の領域が広がっているのです。



そんなこと言ったって
他の部分のことなんて全然
知らないし...

今まで、システムテスト専門というみなさんは、その会社の文化もありつつも、「知識不足で他の領域に手を伸ばせなかった」「知識不足ゆえに手を伸ばそうという考えさえ浮かばなかった」があるのではないのでしょうか？



APIをテスト...！？

聞いたことあるけど絶対難しいやつ！！

わからない！！

ムリムリムリムリ～！！

あなたは最強の武器を手に入れた！



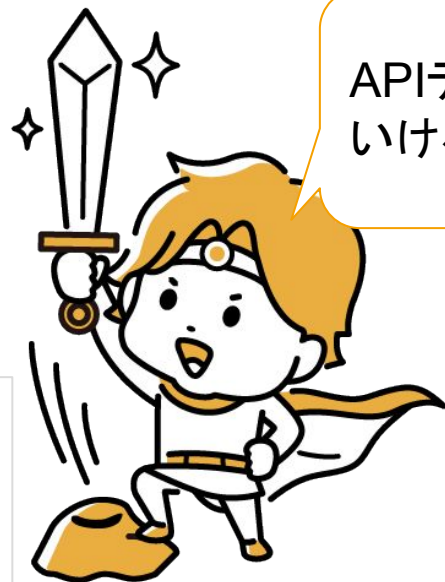
しかし、あなたは今
現代技術を惜しげもなくつぎ込んだ、技術の粋を結集した現代最強の武器を手に入れたのです。

この最強の武器で何をするのもあなた次第！

AIの登場で思い描いていたことができるようになった!

AIの登場により、何かに挑戦する敷居が格段に低くなりました。

なので今までやりたいと思っていたことや、こういったイベントで見て「私もやってみたい!」と思ったことが簡単にできるようになったのです。



APIテストに切り込むぞ!
いけるか!

モチロンです。
APIテストをモノにしま
しょう!

事例：AIを使って冒険をはじめたNさんの例

今まで黒い画面(ターミナル)も見たことがないと言っていたQAのNさん。
AIを手にしたときに、その時困っていた「**テストのやりにくいところ**」を簡易化する
ツールをAIを使って開発してみました。そしたらそれが思った通り動く！
さらに細かな便利ツールを開発、**それらツールをつなげてプロセスも構築したの**
です。
さらに、**AIを使って開発者のコードを読み、それを軸に開発者と「～ってどうなっ**
ているのですか？」と話す ということまで今はしています。
今までやっていた「システムテスト実行」から AIを使い越境したのです。

テストツール
開発

プロセスの
構築

開発のコードを
読み会話



色んなことができるって、何？
～具体例～

色んな事ができるとは言うけど、例えば？

我々は現代最強の武器を手に入れました。

それにより、今まで壁だと思っていた部分を超え、色んな領域に冒険に出られるようになりました。

では、AIを使うことによりどんなことができるのでしょうか？

私が思い浮かぶ具体例をいくつか紹介していきます。

初級

今まで
「大変そう」「面倒だなあ」と思って
なおざりになっていた部分を整えてみよう

私が回ったWebやアプリ系ではあまりテスト計画作成は見かけませんでした。こういったアプローチで今回のテストを行えば効果的・効率的にできるのかを考えることは非常に大切です。

例えば、与えた仕様などからリスクを列挙し、そのリスクを確認するためにどういった**テストのアプローチをすればいいかを AIを使って出す**、といったことも出来るでしょう。

他にも、こういった場所は手動テストで、こういった場所はE2E自動テストで行う、といった**作戦も立てやすくなる** でしょう。

そのテスト項目はどのテスト設計の結果から出たものなのか、そのテスト結果はどのテスト観点から出たものなのか、そのテスト観点はこういった仕様やリスクから出たものなのか……といった**紐づけを行うと、仕様変更などがあってもそこをたどって修正をすることができます。**

理想としてはこういったことをしたほうがわかっていつつも、運用が大変で出来なかったりしました。(私はそう)

そこで、AIを使って一連の紐づけを行い、テストの説明をしやすくする、メンテナンスをしやすくするようなことが考えられます。

勉強会に出たりSNSや記事を見て「そのやり方を試してみたい！」と思いつつも、難しそうで手が出せなかったという経験はみなさんあるかと思います。

そういった**技法を取り入れるのも AIを使うことによりハードルが下がります。**

一例として「テストマトリクスを使ってみよう！」とか「VSTePをやってみよう」(これはちょっとハードルが高いかも...)などです。

社内でのテストは「これが自分たちのやり方だ」が出来上がってきます。
社内の文化やプロセスにフィットしたやり方ですが、実はそもそも基礎を知らずに出来上がった我流だったり、他のやり方を知らない/忘れていた可能性もあります。

そのときに「JSTQBの教科書」(JSTQBのシラバスでもいいけど難しめ)を1周読むと、「欠陥マネジメントをちゃんとできてないよね」など、基本的なプロセスで、やったほうがいいけどやっていなかった場所が見えてきます。

実際に我々も社内で週一で輪読会をしていますが、そこで気づかされることが多いです。

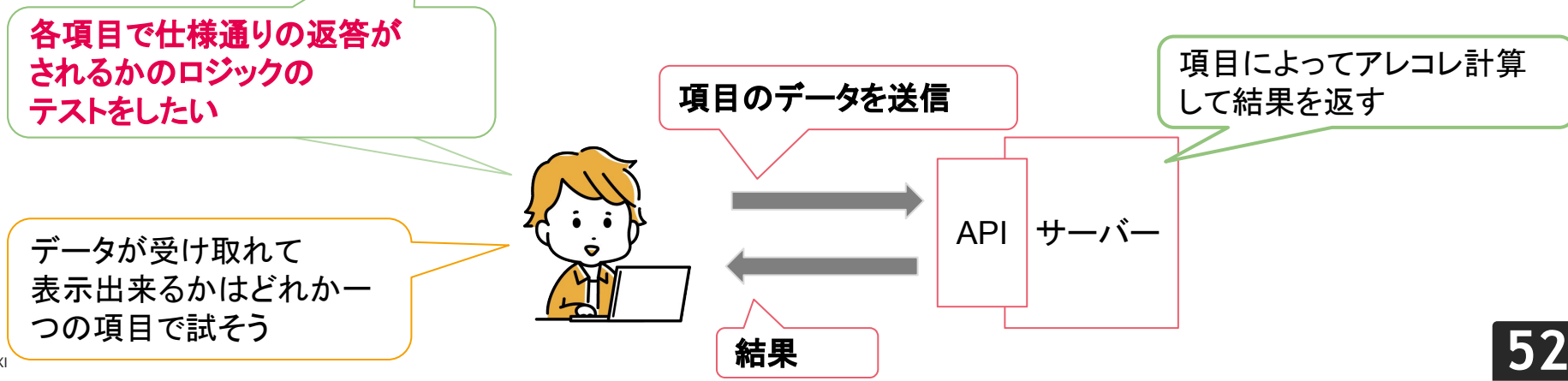
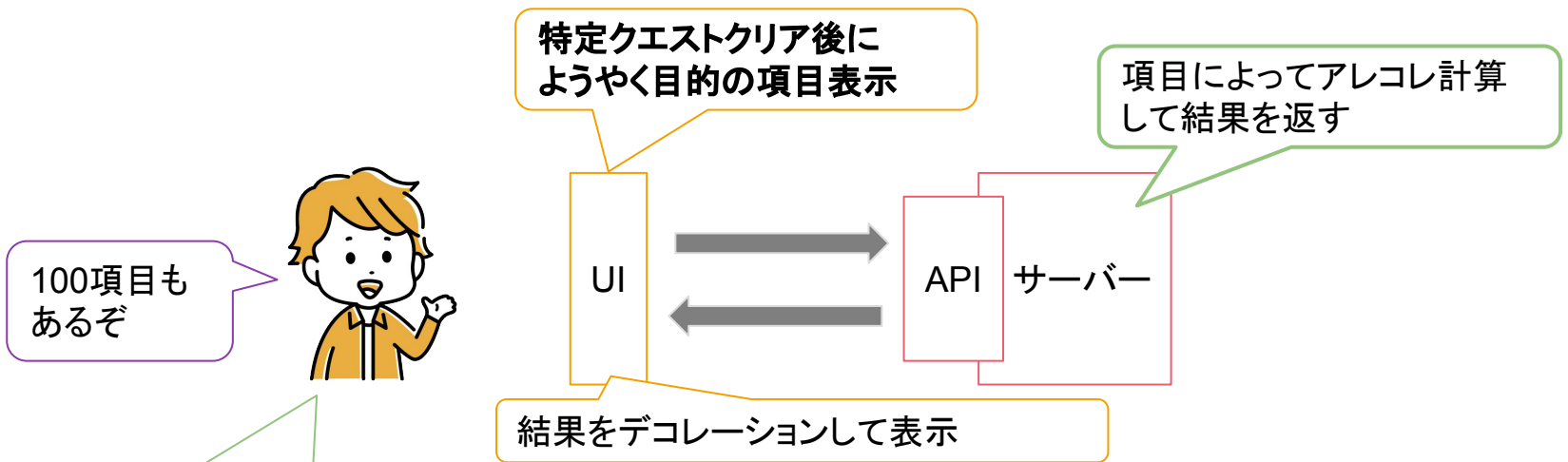
**「やりたかったあれやこれ」
をAIを使って行い
自分たちの今の活動領域を
自分たちの理想形に近づける**

中級

自分たちの今までの領域から
染み出す

今まではUIを通したシステムテストを行っていた人も、AIを使って画面の裏側のAPIのテストに手を出してみるのもよいでしょう。

テストの目的を考えたとき、**UIを通す必要がないものは APIの戻り値で確認できる.....それができるとテストのアプローチの幅が大きく広がります。**



AIを使ってリポジトリにある開発者のテストを確認する こともできます。
ただ「どんなテストをやってるの？」と出しても細かすぎてわからないはずです。
なので「～を確認しているテストはある？」といった聞き方や、「こういうユースケースに当たる部分はテストでどれくらいカバーされている？ 足りなそうなところは？」など、自分たちがわかりやすい粒度で調べたほうがよいです。

AIを使うことで「何か作ること」もとてもハードルが低くなりました。
ユーザーに公開するようなモノを開発するとなると難しいですが、例えばテスト準備を楽にしてくれるツールや、結果を整形してくれるツールなどを作るのは楽になりました。

アイデア次第で、開発者に依頼せずとも様々なテスト効率化ツールを作れるようになったのです。

実際私の周りではこの使い方が多いです。また開発過程で「開発者のテスト」を経験として学ぶ人も多いです。

ツール自体のテストは我々の本職なのでがんばってくださいね！

**AIを使って
今までの活動領域から一歩踏み出し
開拓してみよう**

上級

品質活動を
プロダクトやプロジェクト
全体に広げる

今後皆さんはAIを使って様々なモノを用意すると思います。
テスト観点を生成する、テストケースを生成する、E2E自動テストする、などです。
開発側でもUnit testやIntegration testなどを作っているはずですが。
それら「テストする何か」が点在している状態になります。
それらの便利ツールをつなげ、まるで**ドミノの様に自動的に点在している便利
ツールを実行する「プロセス」を作っていく**ことが今後の活動として考えられます。

これまではシステムテストレベルが活動領域で、システムテストの範囲でのテスト計画を立てている場所も多いかと思います。

ですがAIを使うことでシステムテストレベル以外の部分を把握することもできるようになってきています。

それに伴い、「システムテストレベルでのテスト」から「プロダクト全体を通したテスト」が可能になってきます。

テスト計画も、例えばロジック部分はUnit testでの確認、データ確認はAPIやレビューでの確認、UIを通した確認はE2E自動テスト.....のようなテストレベル全体を通した計画を立てることも考えられます。

開発プロセス全体を通した テストの仕組みの構築



AI時代の タスクのやり方の 再定義

**基本的には
「タスク実行」の部分は
人の手を離れていく**

例外も考えられる

- ・その時のAIでは求める精度に達しない部分
- ・ミッションクリティカルな部分
- ・ゲームやアプリの手触り部分
- ・トークン使用料

人間の動き方はどうなるか？

基本的な流れ

タスク設計



「こういうことをやりたい」という
要求の起点は人。

タスク
実行

AI

AI

AI

AI

タスク実行はAIや機械。

成果物の
妥当性の確認



やった結果、求めていたものになっ
ているのか確認・判断する
のは人。

人間の動き方はどうなるか？ (細かめver)

要求の発生



テスト計画書をさくっと作りたいなー

タスクの方針を決める

リスクベースドでいこう。
E2E自動テストと人でのシステムテストを
計画したい

何をAIで行うか決める

タスクの設計をする

スケジュールは未fixで人が入れるから、リ
スク出しとテストアプローチを今回のAIの
タスクにしよう

タスク実行 (AI)

成果物の妥当性確認

再利用出来るようにするなど完了作業

人間の動き方はどうなるか？ (細かめver)

要求の発生



テスト計画書をさくっと作りたいなー

タスクの方針を決める

何をAIで行うか決める

タスクの設計をする

タスク実行 (AI)

成果物の妥当性確認

再利用出来るようにするなど完了作業

リスク出しはエージェントAを使って、それが終わったらエージェントBでプロダクトリスクに対応するテストアプローチを記載とするか。
必要となるドキュメントも集めない。

リスクが十分に挙げられて、それを確認する
テストアプローチも書けているな

繰り返し使うしチームで使うからSkill化しておこう

我々が要求を出し、AIがタスク実行した結果、それが**我々が思い描いた要求通りになっているかは確認しなければわかりません。**

重要なこととして「**人間が正しく要求を出せているか**」ということもあります（「顧客が本当に必要だったもの」で検索）

超ゴージャスなチョコパフェが食べたい！

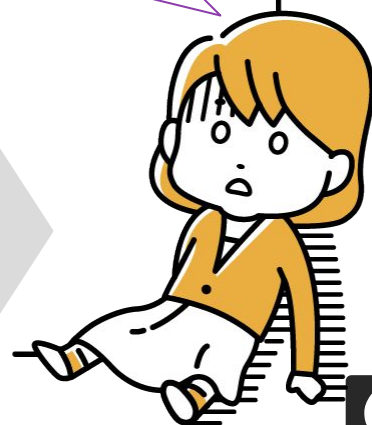
要求



成果物

へい、お待ち！
**高級チョコソースと
こぼれるほどのイクラを
のせた超高級パフェ！**
(AIによる要求の解釈)

違う...
そうじゃない...



これはGPT-5.5で本当にあったお話です.....。

ある日、私はAIを使ってテスト観点出し→テスト設計→テスト項目をまとめて出そうとしたのです。

仕様書や使うデータを用意して、プロンプトを書いてAIを実行しました。

そしたらモノの数分で成果物が出来たじゃありませんか。

私は「早いなー早いなー」と思いながら成果物をチラッと見ました。

そしたらですよ！

テスト項目が.....5件しかないのです！！

私は恐る恐るAIに聞きました。「なんで.....？」って。

するとAIは言ったのです。

****おっしゃる通りです。この規模にしてはテスト項目が少ないですね****

私は趣味で小説を書きます。ある時AIを使って文庫1冊程度(10章構成くらい)の文章を作ってみました。それを楽しみながら読んでいたのですが、途中でこんな表現が.....

「回復アイテムもないんだ。諦めて地上へ戻るしかない」
彼は**第5章と同じ表情**を浮かべていた。

「AIが作ったものにレビューはいるのか」という話を聞くようになりました。レビュー工数増大によるものです。AIはちょっとしたミスをすることもあります。それよりつらいのが、時折人間では起こらないような想像を超えた謎のミスをすることがあります。

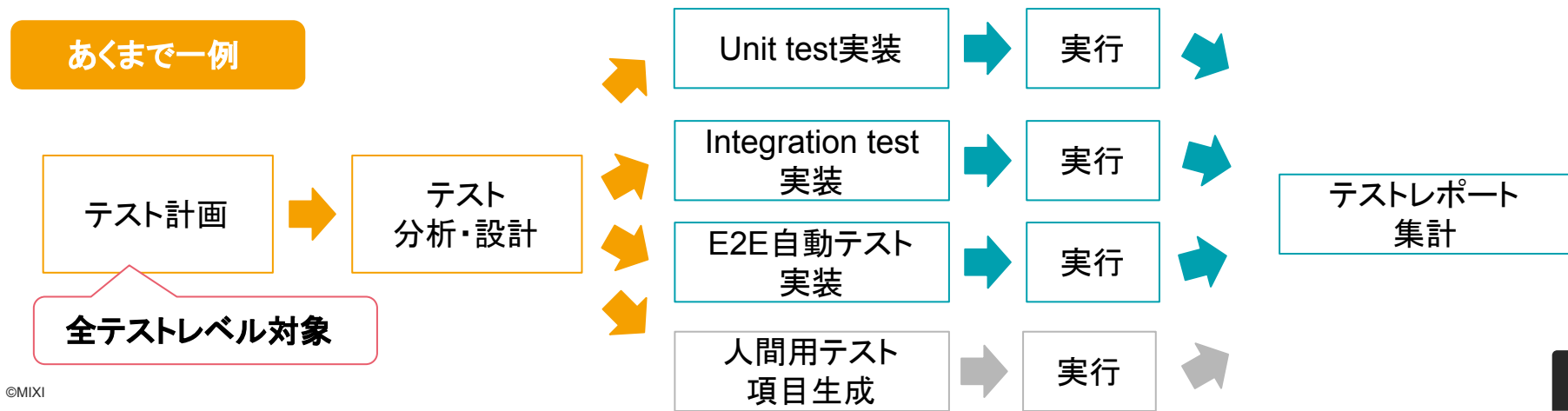
人間のレビューか、何かしらのレビュー機構は必要です。



これからQA/テストエンジニアは
どうなっていくのか？
～ 役割の再定義 ～

我々はシステムテストレベルだけではなく、開発プロセス全体での活動が出来るようになります。

開発プロセス全体を通したテストについてのエンジニアリング、つまり**開発プロセス全体を一気通貫でテストをする仕組み (システム)やパイプライン構築が出来ること**——これは組織が喉から手が出るほど欲しがります。



**テスト活動にフォーカスして
開発プロセス全体のテスト活動を
整備・仕組み化する**



テストエンジニア

品質保証活動はPJメンバー全員で意識し行うものです。
しかしPJメンバー全員が品質保証に明るいわけではありません。
よって、PJが狙ったものを狙った通り作るためにも、**モノづくりのルールやみんなの動き方(プロセス)、AIの使い方やガードレール設定**といった「**良いものを作る環境**」を整備する必要があります。
また、**メンバーへの品質保証活動の教育・育成**を行うことも重要です。
よくある釣りの例えだと、「魚」を与えるのではなく、「釣り場」を整え「釣り方」を伝えてメンバーが自分たちで魚を獲れるようにするといったイメージです。

人・AIを含めた
PJのルール・プロセスの
設定

品質保証観点での
メンバーの
教育・育成

品質保証活動にフォーカスして
環境(ルール・プロセス)整備と教育を行い
PJ全体で品質保証活動できるようにする



QAエンジニア



**多くの活動がAIを使うことによって
なくなった時代**

AI使用は人が起点と言いましたが、テスト設計をAIで行う...テストプロセス全体をAIで行う...全体のプロセス整備をAIで行う...というように、AIだけで良しの部分が増え、起点はどんどん上がっていくことが予想されます。

AIで、こういった**作業が削ぎ落されていった結果、原初の問いに行きつきます。**

我々が実現したいことは何か？



**ユーザーを
満足させたい！**

実現したいことは変わらない。実現手段が変わっていく

この「ユーザーを満足させたい！」は根源であり、これまでも、そしてAI時代になってからも変わることはありません。

これまではそれを手動テストで実現していましたが、昨今ではAIを使って実現しています。

我々が実現したいことは変わることはなく、実現手段が変わるのです。

昔

ユーザーを満足させたい！



クリティカルなバグはない
方が満足できるよね



手動テスト

今

ユーザーを満足させたい！



クリティカルなバグはない
方が満足できるよね



AIを使ったテスト

将来

ユーザーを満足させたい！



みなさん次第で出来ることは無限大！

ユーザー満足を実現する
仕組みを作る



AI時代における テストの基礎の再定義

テスト = 開発プロセス全体のテスト活動

現在のQA/テスト業界の暗黙の了解として「テスト」というと基本的にはシステムテストレベル以上の話であることが多いです。

しかしAIを使うことによって、今まで尻込みしていた場所に手を伸ばせるようになったのです。システムテストレベルだけのテストから飛び出すことができます。

テスト = 開発プロセス全体のテスト活動として捉え直すと視野も活動領域も広がります。

テストにまつわる知識を 学ぶ必要がある

作業するのはAIだとしても「**こういうテストがしたい**」や「**こういうテストの仕組みを作りたい**」という最初の起点 (要求) はあなたが出す必要 があります。これはあなたの中にその「**発想ができる種 (知識)**」がないと芽生えません。
例えば「APIテスト」を知らなければ「APIの戻り値を確認して担保しよう」という発想に至ることはありません。もちろんAIにその指示を出すこともできません。

テストを学び 成果物を判断 (評価) できないといけない

AIという便利な道具を手にしたところで、使うのはあなた です。
知識がなければ、AIの出力の妥当性の判断はできません。照らし合わせるベースラインが頭の中にないためです。結果「AIの出力は全て合っている」という思い込みも発生します。
テストを知らない人ほど「これでいいじゃん?」「レビューなんている?」となっているのを見てきています。
自分の中に良し悪しがわかる判断材料を用意する必要 があります。

AIに自分の意図通りの指示が出来る力 AIの回答を読解する力 ～ 国語力 ～

AIが指示通り動いてくれないという人のプロンプトを見ると、曖昧な書き方をしていることが多いです。

日本語は主語や目的語が抜けたりと、構造の自由度が大きい言語です。抜けによって意図がAIに伝わらないことがあります。

またAIからの回答を読解できないと、AIの間違いや自分の考えとの齟齬に気づけません。文章を正しく読み取る力も必要です。

AIを使いこなせる力

今後、より一層AIを使った業務が増えていきます。
これまでの「パソコンが使えること」や「Officeが使えること」レベルで当たり前に見えることを求められるようになっていくと思われます。
先の国語力はもちろん、**今のAIツールではどういったことが出来るかの基礎は押さえておいたほうが良い** でしょう。

Don't pave the cow path.
“牛が踏み固めてできた道を、そのまま舗装するな ”

牛が何度も歩いた結果、自然にできた曲がりくねった道を「昔から使われているから」という理由で、そのまま立派な道路にしてしまうイメージです。

古いやり方や非効率なテストを、見直さずにそのまま自動化や AI化してはいけません。

今はまさに転換期です。既存のやり方を、**今のやり方に適したプロセスに変えられる力が必要**です。



AI時代のQA/テストエンジニアが 学ぶこと

AIの基礎知識

テストの基礎知識

開発の基礎知識

**開発プロセス全体を通した
仕組み化をするための知識**



AIの基礎知識

～ 重要部分ピックアップ ～

会社で使っている AIを知る (情報漏洩、セキュリティ)

CodexやClaudeなど、現在は様々なAIツールがあります。
便利だからといって自由に選ぶのではなく、**会社で使用を認められている AIだけを使しましょう。**

入力した情報が学習に利用されるか、どのくらい保存されるか、誰がアクセスできるかは、サービスの契約や設定によって異なります。

機密情報、個人情報、顧客情報などを入力する前に、必ず社内ルールと利用条件を確認してください。

未承認のツールに業務情報を入力すると、情報漏洩につながる 可能性があります。

AIが得意なこと、苦手なことを知る

AIは、文章の作成や要約、情報の整理、アイデア出し、プログラムの作成などを得意としています。

一方で、暗黙のルール、最新情報、正確性が必要な判断などは苦手です。
また、AIは自信があるように間違った回答をすることがあります。
大量の内容を一度に作らせたときに、一部が抜け落ちることもあります。

AIの得意・不得意を理解し、**どこまでをAIに任せ、どこからを人が確認するのかを判断できることが大切**です。

コンテキストの与え方を知る

AIの出力は、指示の書き方だけでなく、どのような背景情報（コンテキスト）を与えたかによって大きく変わります。

目的、対象、前提条件、制約、出力形式、参考資料、過去の事例などを与えることで、AIはより状況に合った回答を出せるようになります。

逆に、必要な情報を与えずに「いい感じに作って」と頼んでも、期待した結果にはなりにくいです。

単にプロンプトのテクニックを覚えるのではなく、AIを使って作業するために必要な情報を整理し、過不足なく渡せるようになることが重要 です。

モデルを知る

AIツールには複数のモデルがあり、それぞれ性能、速度、利用料金、得意な作業などが異なります。

常に一番賢いモデルを使えばよいわけではありません。高性能なモデルほど、利用料金や処理時間が大きくなる場合があります。

文章の整形や簡単な分類などには軽量なモデルを使い、複雑な分析や設計には高性能なモデルを使うなど、作業に応じた使い分けが必要です。

モデルは頻繁に変わるため、すべてを細かく覚える必要はありません。

どの程度の作業に、どの程度のモデルが必要なのかを試しながら把握し、品質とコストのバランスを考えられる ようになりましょう。

ツールの機能を知る

CodexやClaudeなどのAIツールは様々ありますが、できることは似通っています。重要な機能は把握しておくといいです。

Skills... 「こういう処理をして」という作業指示書みたいなもの。定型作業の自動化の他、「テスト観点 skill」などを作り、自分たちのやり方を他チームに渡すような使い方が可能

プランモード... 作業を始める前にどのように作業するか計画を立てるモード。作業前に AIがどう作業しようとしているかがわかる

外部ツール連携(コネクタ/プラグイン/MCP)... 外部ツールとの接続を行う。例えば Notionとつないで、その内容をコンテキストとして読み込むなどが可能になる



テストの基礎知識 ～ 重要部分ピックアップ ～

テストの一般用語を知る

前に探索的テストのことを「ウォークスルー」と呼んでいた現場がありました。一般的なテスト用語ではウォークスルーはレビュー方法です。

こういった現場でのみ通じる言葉でAIに「ウォークスルーのやり方を教えて」といっても正しい回答を導けません。

まずは、**一般的に通じるテストの共通語を覚えること** がAIを使うためにも優先です。

オススメ本

JSTQBのシラバス(無料)

ソフトウェアテスト教科書 JSTQB Foundation 第5版 シラバス2023対応
大西 建児 (著), 湯本 剛 (著), 町田 欣史 (著), 福田 里奈 (著), 角田 俊 (著), 藤原 考功 (著), 大段 智広 (著)

テストを体系的に知る

テストと言っても様々な活動があることを先に示しました。

こういった活動があるのか、その活動はこういったことを指しているのかを知る必要があります。そうしないとAIで「仕様通り動くか確認するためのテストケース作って」という状態になってしまいます。

また**知ること自分たちの立ち位置、足りないもの、余計なものも見えてきます。**

オススメ本

JSTQBのシラバス(無料)

ソフトウェアテスト教科書 JSTQB Foundation 第5版 シラバス2023対応
大西 建児 (著), 湯本 剛 (著), 町田 欣史 (著), 福田 里奈 (著), 角田 俊 (著), 藤原 考功 (著), 大段 智広 (著)

テスト設計を学ぶ・使う

「仕様通り動くか」だけではテストが足りません。境界値だったときは問題がないのか、状態の遷移はひとつだけでいいのかなど、**仕様からは見えないパターンまでテストをする設計していく必要があります**。AIも出せますが、レビューをすると明らかにパターン不足の場合もあります。**AIの成果物評価が出来るためにも**、テスト設計を理解できるように学び、実際に手を動かして把握しておく必要があります。

オススメ本

[改訂新版] マインドマップから始めるソフトウェアテスト

池田 暁 (著), 鈴木 三紀夫 (著)

ソフトウェアテスト技法練習帳 ~知識を経験に変える40問~

梅津 正洋 (著), 竹内 亜未 (著), 伊藤 由貴 (著), 浦山 さつき (著), 佐々木 千絵美 (著), 高橋 理 (著), 武田 春恵 (著), 根本 紀之 (著), 藤沢 耕助 (著), 真鍋 俊之 (著), 山岡 悠 (著), 吉田 直史 (著)



開発の基礎知識 ～ 重要部分ピックアップ ～

プログラミングの基礎を知る

今のテストの領域を広げるためにもプログラミングの基礎を知っておくのは良いです。メソッドが集まってプログラムは構成されている、などです。

本などで読んでも難しいので、**AIを使って小さなツールやゲームを開発してみる**のが早いです。

開発した後に一つずつ AIに質問してどう動いているか把握すると良いです。いくら質問しても怒らないので理解できるまで質問してみましょう。

オススメ本

すぐわかる！ ふよふよプログラミング SEGA公式ガイドブック

紀平 拓男 (著)

Gitの基礎を知る

今のプログラムの管理はほとんどの組織がGitを使って行われます。

これを知ること、開発側との協業もやりやすくなります。

プログラム作成練習を行ったら、Githubに保存するといった練習を行うとよいです。また自分で作ったものならいくらでも失敗ができます。様々試してみましょう。ここもAIを通して操作可能ですし、AIが詳しい領域です。わからないことは事細かに聞いて覚えるとよいです。

オススメ本

改訂2版 わかばちゃんと学ぶ Git使い方入門

湊川あい(著), DQNEO(著)

Unit testの基礎を知る

恐らくUnit testの存在は知っていても、どんなテストなのか、システムテストと何が違うのか知らない人もいます。

知ることによって、こういった場面で Unit testが有効なのかという理解も進みます。

実際に作ったプログラムに対して、AIを使ってUnit testを書いてみると良いです。またAIに詳細に話を聞いて把握するようにすると勉強になります。

日本語が使える言語なら、テストメソッド名を「何をテストするか」(テスト意図)を日本語で記載するとわかりやすいです。

オススメ本

単体テストの考え方/使い方
Vladimir Khorikov (著), 須田智之 (翻訳)

Webアプリの基礎を知る

今扱っているモノのほとんどが、クライアントがAPIを通してサーバーにアクセスし、サーバーがDBを読み書きするモノかと思います。

その一連の流れをおおよそ知っておくと、今後こういったテストが必要なのかを知るきっかけとなります。

ここもAIを使って自作してみることをお勧めします。AWSなどは初心者が使うと課金やセキュリティ周りで危ないので、ローカルで完結するような、ちょっとした保存をする程度の小さな構成を作ってみましょう。FastAPI, SQLiteなどが使えます。

DB, APIの基礎を知る

APIのリクエストとレスポンス、HTTPメソッドやステータスコード、DBの検索・登録・更新・削除といった基礎を知ることによって、画面の裏側で何が起きているのかを理解できます。

その一連の流れがわかると、どこで不具合が起こり得るのか、どの層をどのようにテストすべきかを考えられるようになります。

これもローカルで動かせる小さなWebアプリをAIを使って開発し、API、DBの処理を詳しくAIに聞いて学ぶのが良さそうです。



開発プロセス全体を通した 仕組み化をするための知識

プロダクト全体のテストを知る

プログラミングや DB、APIを学ぶと、システムテストだけでは見えなかった必要なテストが見えてきます。

システムテストの観点だけではなく、一連の開発を通して見える、こういったところにこういったテストが必要なのかを学び、考えましょう。

自分のローカルでWebアプリを作ったら、そのWebアプリを公開するようなつもりで、開発プロセス全体でのテスト計画、テスト分析、テスト設計、テスト実装、テスト実行、テストレポートの練習をしてみるとよいです。

仕組み化の方法を知る

今後AIを使って様々なテストを出来る様にしたり、品質保証活動の効率化をしていくことになります。

ただ、それが**局所的に点在しているだけでは、上手に活用されない**ことが考えられます。

点と点をつないで線にする、つまり、必要なテストや品質保証活動が継続的に実行される仕組みにする必要があります。こういった仕組みやパイプラインを作るための知識をつけておくとプロセスづくりに活用できます。

一例としてはGitHub Actionsがあります。勉強方法として自分が作ったアプリに対して一連のテストが動くようなGitHub Actionsを組んでみるのもよいでしょう。



まとめ

**AIは道具である。
道具である以上、起点は人である。
目的を果たすため、人が道具を上手に使う。**

**奪われたのは仕事そのものではない。
一部のタスク実行工程である。**

**あなたは最強の武器を手に入れました！
広い品質の世界のどこにだって行けます。
あとはあなたがやりたいこと次第です。**

テストエンジニア

**テスト活動にフォーカスして
開発プロセス全体のテスト活動を整備・仕組み化する**

QAエンジニア

**品質保証活動にフォーカスして
環境(ルール・プロセス)整備と教育を行い
PJ全体で品質保証活動できるようにする**

我々の仕事は...

ユーザー満足を実現する仕組みを作ること

テスト = 開発プロセス全体のテスト活動

テストの基礎知識を身に付けること

成果物の妥当性を評価できること

意図通り伝える / 受け取るための国語力

AIを使いこなせること

今に適したプロセスに変えていくこと



我々が
実現したいことは何も変わっていない。
変わるのは実現手段である。

心もつなごう。

MIXI



Appendix①

AI時代でも勉強は必要

AI時代でも勉強は必要

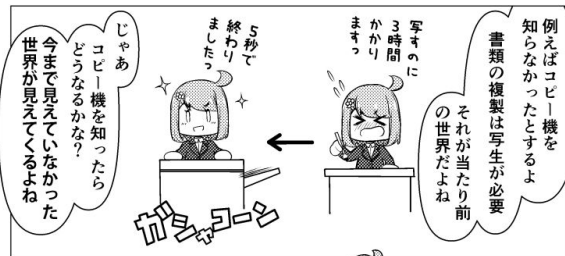
AI時代だし、勉強しなくてもAIが教えてくれる...
そう思っている方もいるのではないのでしょうか？
しかし大事なこととして、「知らないことは AIに
伝えることができない / AIに聞くことができない」
ということがあります。

なぜなら知らないからです。あなたの世界にその
言葉、その概念がないからです。

よって聞くことも近づくこともできないのです。

**知識によって、あなたが見ることができる世界
が広がるのです。**

箸休め 勉強するとどうなるの？

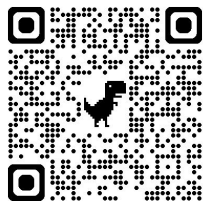


Appendix②

AIがテスト工程を通して実行する仕組み



テスト計画・分析・設計・実装・実行・ レポートまでを通して自動実行する Skillを開発した話



<https://zenn.dev/jam0824/articles/c84a2ef393ee70>

問題

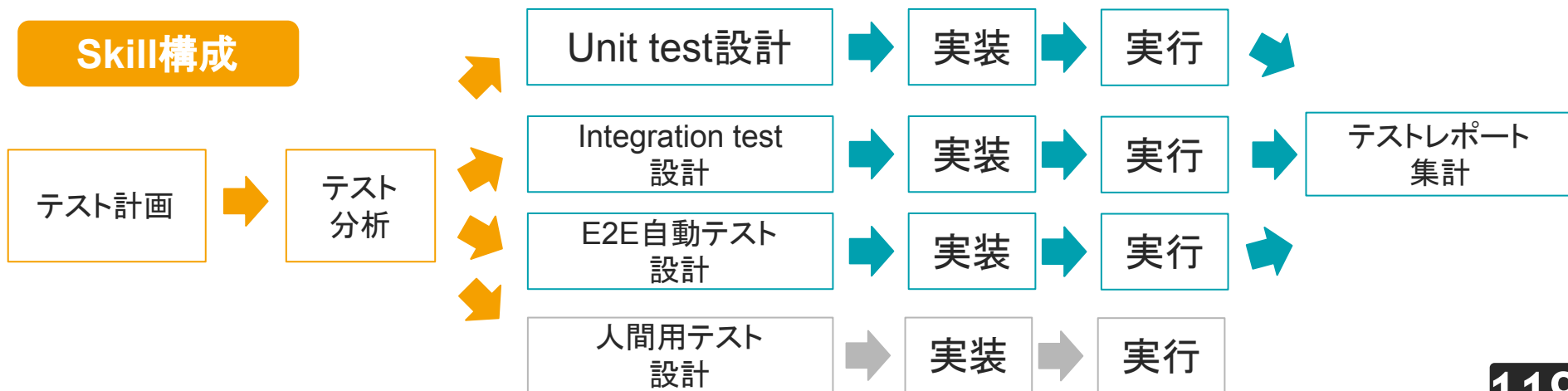
いきなりテストケースを作ると
「そのテストケースはなぜ必要か」と
「何が抜けているか」を追えない

計画からレポートまでを全自動で実行するSkill

各Skillのオーケストレーションにより**計画から分析・設計・実装・実行・レポートまでを全自動実行**します。

Unit／Integration／E2E／人間実行を分け、**中間成果物を作り**次工程に渡しながら進みます。

リスク→観点→設計→項目と**追えるように紐づけている**ことが特徴です。



長い工程ほど、工程ごとにレビューを閉じる

開始条件と入力

工程ごとに成果物とレビューを閉じる

方針・前提を確認

中間成果物を作成

問題に優先度を付けてレビュー

計画・分析・設計・ケース作成の各工程で「成果物 → レビュー → 修正」を優先度が高い修正がなくなるまで繰り返す。

大きな問題を修正

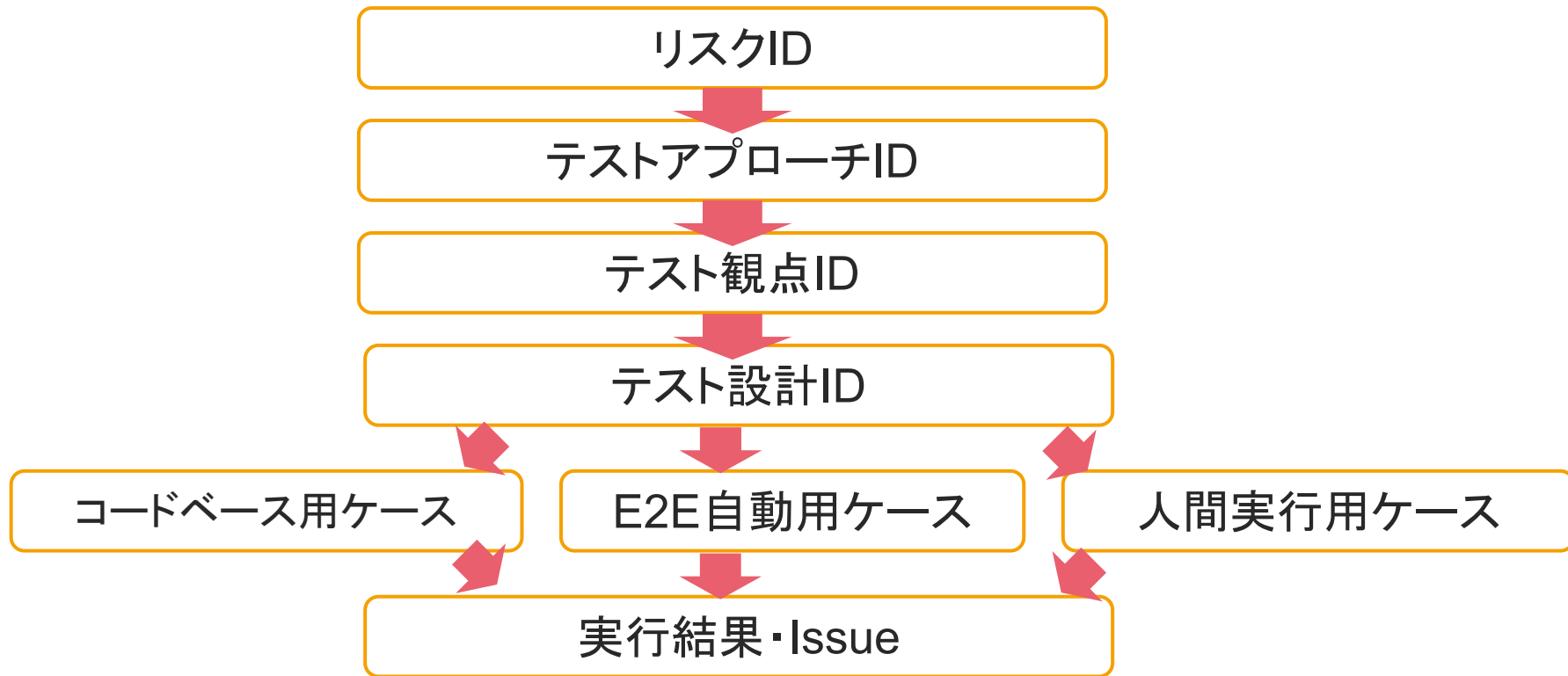
大きな問題がなくなったか確認

不明点は質問票として書き出し、AIの推測で埋めない。

次工程へ引き渡す

単純な連続実行ではなく、工程ごとにスレッドを分けて丁寧に進める。

IDの連鎖が、テストケースの「理由」を残す



トレーサビリティを確立し
仕様変更時も、IDをたどって影響範囲を追える

テストケースは、実行主体ごとに分けて出力する

コードベース用

Unit／Integrationのテストコードを生成・レビュー・実行

E2E自動用

PlaywrightでUIを通すテストを生成・レビュー・実行

人間用

機械で判断できないもの(人が実行した方が良いもの)

質問待ち

不明点、未実装は質問を行う。推測でケースにしない。

最低限の入力は、仕様とコードの場所

過去バグやプロジェクト固有の観点も、追加の入力として渡せます。

入力: 仕様、プロダクションコード、任意の追加情報

実行: `run-test-process` が工程ごとにskillを呼び出す

目安: サンプルでは全工程が約1時間15分。工程ごとに丁寧に進める

※リスクやテスト観点などは全て仕様などのテストベースから導かれる。その段階ではコードは使われない。
実装時にコードベースが必要なためコードの場所が必要

実行結果は、未完了も含めて見える形に戻す

1. テスト結果サマリー

全テストケース数

215

実行済みテストケース数

174

実行率

80.9%

Pass

161

Pass率

92.5%

Fail

13

N/A

0

未実行（実行対象）

32

質問待ち

9

未実装

1

Pass率 = Pass / (Pass + Fail)。N/Aは分母に含めない。

未実行（実行対象）は、質問待ちと未実装を除いた未実行ケース数。

記事掲載の実例

AI側で完了した結果だけでなく、

- ・**実行率／Pass／Fail**
- ・未実行／質問待ち／未実装
- ・コードベース／E2EのIssue

を一画面にまとめます。

「完了」ではなく、人が次を判断する入口です。

AIは
トレーサビリティと中間成果物の維持を
現実的にする。

Zenn: zenn.dev/jam0824/articles/c84a2ef393ee70

GitHub: github.com/jam0824/ai_test_process_skills

心もつなごう。

MIXI

