ITエンジニアの救世主？
～生成AIで変わるシステム開発自動化の未来～

2024年度 アシストソリューション研究会
東日本「システム開発自動化の未来」分科会

「システム開発自動化の未来」 メンバー紹介

氏名	役割	所属機関・企業
小平 綾香	リーダー	エムアンドシーシステム
外塚 雄也	サブリーダー	NTTコムウェア
小西 航太	サブリーダー	NTTコムウェア
松本 翔	メンバー	JR東日本情報システム
湯浅 勇貴	メンバー	日本精工
小池 匡紀	メンバー	シー・アイ・シー
宮田 治	メンバー	メディカルシステム研究所
栗原 一樹	メンバー	東京ガスエンジニアリングソリューションズ
池田 拓海	メンバー	芝浦機械
伊東 清音	メンバー	アシスト
河井 亮	メンバー	アシスト

システム開発とは

- 目的・ニーズを明確化し、ソフトウェアの作成・運用を行うプロセス
- 代表的な開発手法のWF（ウォーターフォール）開発は次の工程に分かれる

代表的な工程	概要説明
要件定義	システムに必要な要件を明確化
設計	システムの構造や技術仕様を設計
実装	システムを構築
テスト	システムが要件を満たしているか、不具合がないかを確認
デプロイ	システムを正式に稼働
運用と保守	定期的な監視と改善

システム開発自動化の現状①

代表的な工程	現状
要件定義	**自動化はまだ進んでいない**
設計	
実装	コーディング　：生成AIを用いた自動化が進んでいる コードレビュー　：**自動化はまだ進んでいない**
テスト	計画、準備、設計：**自動化はまだ進んでいない** 実装、実行　：生成AIを用いた自動化が進んでいる
デプロイ	専用ツール導入による自動化が既に進んでいる
運用と保守	

システム開発自動化の現状②

- 生成AIによる開発支援ツールの普及率の予測(Gartner社の調査)
  2023年　**約10%**　→　2028年　約75%

- **一部の企業から、システム開発の生成AIの活用を開始**

I社
ツールによるコード自動生成

C社
GitHub Copilotを全社的に導入

生成AIのリスキリング

L社
社内全エンジニアを対象に
GitHub Copilot を導入

参考
株式会社日本総合研究所 先端技術ラボ、生成AIを活用したシステム開発の現状と展望- 生成AI時代を見据えたシステム開発に向けて -
https://www.jri.co.jp/MediaLibrary/file/advanced/advanced-technology/pdf/15272.pdf

システム開発の問題

工程	主要な問題
要件定義	正確性の維持が困難、統一性の無さ
設計	
実装	属人性あり、生産性の低さ
テスト	工数大、前工程の遅れの影響大、属人性あり
デプロイ	属人性あり、生産性の低さ
運用と保守	

研究の目的

生成AI活用によるシステム開発の問題解決に向けた示唆を提供

ドキュメント作成自動化

生成AIが、仕様書や設計書などを自動生成または補助できるか検討

コーディング自動化

生成AIによって、ソースコードを効率的に作成する方法を検討

テスト設計自動化

生成AIがドキュメントを元に、テスト設計を自動化できるか検討

「システム開発自動化の未来」の研究における定義

観点	定義
未来	2024年12月現在より先の将来
自動化	半自動化＝生成AI技術を活用して開発作業の一部を効率化 生成物の妥当性や品質の判断などは人間による実施が必要
訴求先	システム開発に携わる人(開発者) 開発者が直面する課題を解決するための技術的示唆を提供したいため、 初心者を除く

製品選定

GitHub Copilot / ChatGPT *

システム開発支援に特化

コーディングの生産性と効率の向上を
目的としている

日本語への最適化

OpenAI社は東京オフィスを開設
→日本語への最適化が期待できる

多くの言語に対応

Python、JavaScript、
Ruby、C#、C++、Java
etc.

業界内で実績あり

一部企業が利用中

*共にOpenAIのLLM(大規模言語モデル)であるGPTシリーズを利用

ドキュメント作成自動化

開発におけるドキュメントの主要な課題

主要な課題	補足
コスト	・ドキュメント作成、更新、レビューを行うために人手・時間を要する
組織・文化	・属人化により、ドキュメントが存在しない ・設計書やテスト仕様書といったドキュメントを残す文化が希薄
統一性	・独自フォーマット・記法の設計書の要求 ・人によって記載粒度や異なる

生成AIを用いて、ドキュメント作成がどこまで効率化できるのか検証

検証概要

以下をポイントとして、検証を実施

①生成AI利用による、既存文書からの後続ドキュメント作成の可否・比較
②既存コードからドキュメントへのリバースエンジニアリングの可否

参考：使用した文書とソースコード

研究を効率的に進めるため、インプットデータはインターネット上に公開されているドキュメント、ソースコードを活用

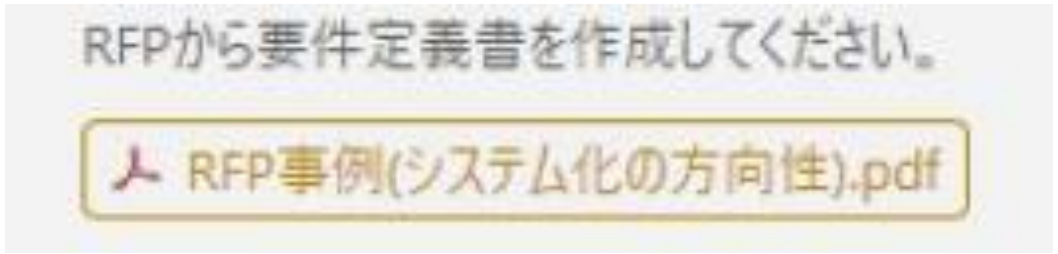
インプットデータ	概要
ドキュメント	防災科学技術研究所のeコミュニティ・プラットフォームの システム概要書、操作説明書
ソースコード	ソーシャルオープンプラットフォーム利用推進フォーラム (SOPPF)の 相互運用gサーバーのJavaファイル(約200ファイル)

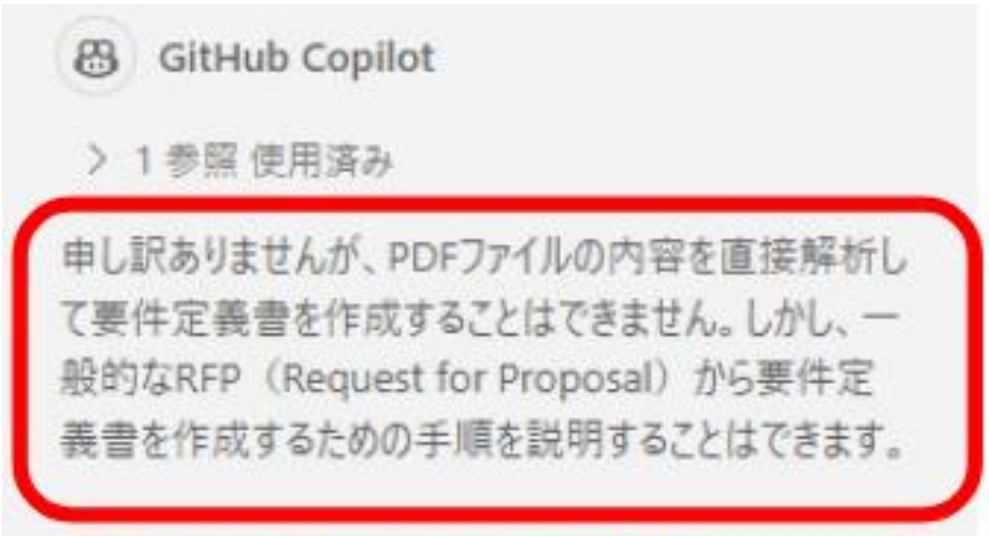
利用したツール

ドキュメント作成においてはChatGPTを採用

ツール	補足
ChatGPT	有償のPlusプラン（主にChatGPT 4oを使用）
GitHub Copilot*	Copilot Businessサブスクリプション

*2024年10月時点では、
GitHub Copilot がドキュメントの読込不可



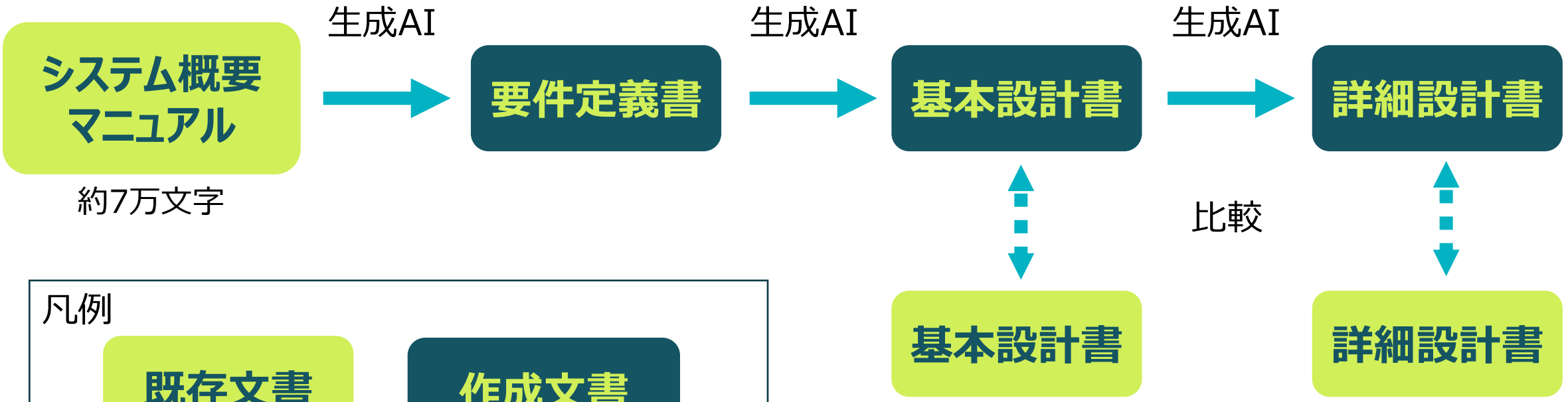


検証パターン概要

検証パターン	内容
パターン①	システム概要・マニュアルから設計書を作成し、既存設計書と比較
パターン②	ソースコードから設計書を作成 (リバースエンジニアリング)

検証パターン①の概要

システム概要・マニュアルから設計書を作成し、既存設計書と比較



検証パターン①の結果

1. 既存の概要・マニュアルから、要件定義書の作成が可能
   ⇒見出し・概要レベルの文書が自動生成
2. 既存文書との比較：見出しレベルでは合致、情報は網羅できず素案レベル

	既存文書			作成文書	
	ページ数	文字数		ページ数	文字数
概要・マニュアル	185	73,391	1.自動生成	-	-
要件定義書	-	-		4	2,520
基本設計書	73	34,372		4	3,475
詳細設計書	125	66,296	2.比較	5	4,736

検証パターン① 検証イメージ（1/2）

①AIにシステム概要、マニュアルを読み込み

インプット

システム概要

概要

地域住民自らが参加型で地図を作成し、グループ内や外との情報共有が行えるWebマッピングシステム（Web-GIS）です。当研究所が開発した、地域協働を支える情報基盤「ｅコミュニティ・プラットフォーム（ｅコミ）」の主要アプリケーションあり、サイト構築・運営ツールであるeコミグループウェアと一体的に利用することができます。本システムはオープンソースにより無償公開しています。

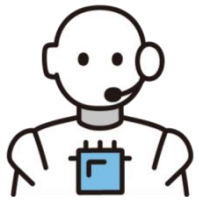
マニュアル

① 表入力インターフェイス

登録情報項目の全レコードが表示されます。

セルをクリックするか、キーボードのカーソル移動をすることで、直接編集ができます。編集内容を確定する場合は、Enter キーを押します。

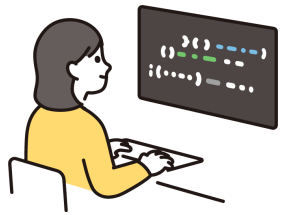
列名をクリックすると、その値で並び変えた結果が表示されます。

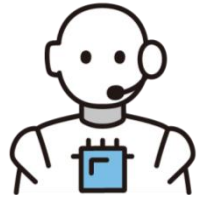


②要件定義書、設計書の作成を指示

指示

これまでアップしたファイルの内容とあなたの分析結果をもとにして、eコミマップの要件定義書を作成してください。要件定義書としてeコミマップの機能を詳細、網羅的に記述した内容で作成をお願いいたします。





検証パターン① 検証イメージ（2/2）

③AIからアウトプットを受領

素案レベルのドキュメントを生成可能

導入部

eコミマップ要件定義書↵

1. はじめに↵

1.1 背景↵

eコミマップは、地域住民や自治体が共同で地図を作成し、災害情報や地域情報を管理・共有できるWebベースのGISシステムです。本システムは、地域社会の協力を促進し、災害対応や平時の地域情報管理に利用されることを目的としています。↵

1.2 目的↵

この要件定義書は、eコミマップシステムの機能、技術的要件、運用要件を詳細に記述し、システム開発の基盤を提供します。↵

運用要件

4. 運用要件↵

4.1 ユーザーサポート↵

- **管理者向けドキュメント**: eコミマップの設定や管理に関する操作説明書を提供し、システム管理者が円滑にシステムを運用できるよう支援。↵
- **一般ユーザー向けガイド**: 初めて利用するユーザー向けに、基本的な操作方法やトラブルシューティングを含むユーザーマニュアルを提供。↵

4.2 メンテナンス↵

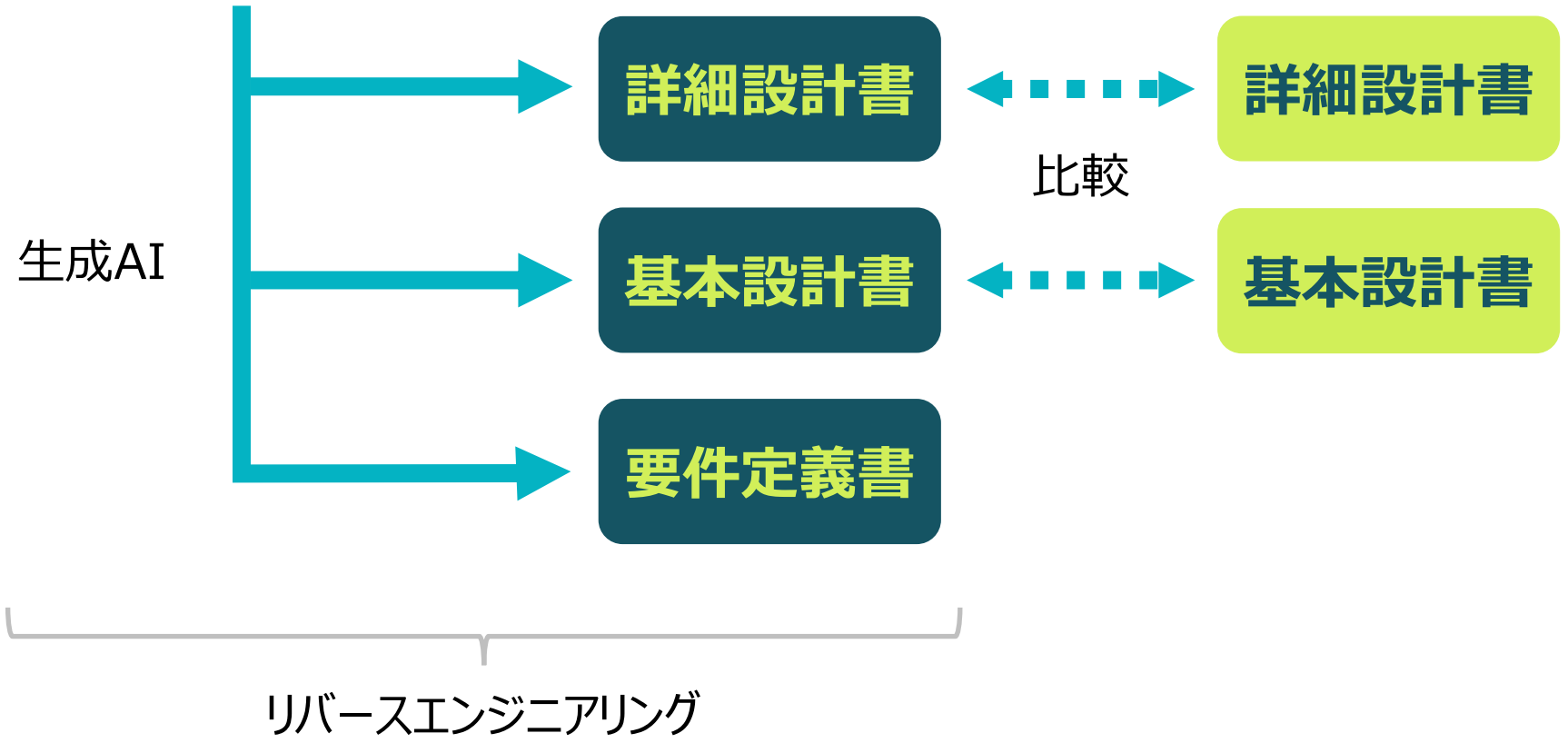
- **定期的なバックアップ**: システムのデータは定期的にバックアップされ、災害時などの緊急時にも復元可能な体制を整備。↵

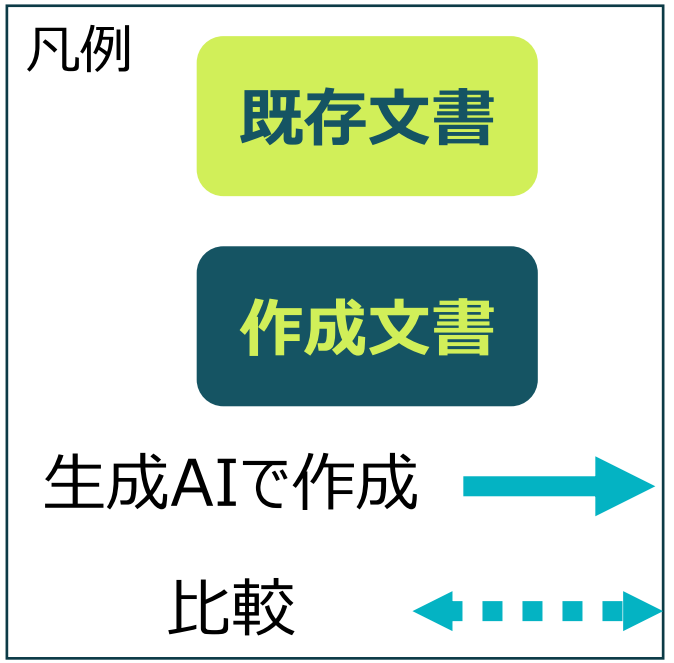
検証パターン②の概要

ソースコードから設計書を作成 (リバースエンジニアリング)

ソースコード

地図情報登録・配信システムのソースコード
(約200ファイル)





検証パターン②の結果

- インプット量のキャパシティ超過のため、文書ファイルの生成不可
- システムの目的や構成に関してChatGPTに更問した結果、**システム概要をテキスト表示することは可能**

	既存文書		作成文書	
	ページ数/ファイル	文字数/行数	ページ数	文字数
ソースコード	192 (ファイル)	54,819 (行)	-	-
詳細設計書	69	61,287	ファイル出力不可 (システム概要のテキスト表示は可)	
基本設計書	77	27,170		
要件定義書	-	-		

補足：検証パターン②の出力イメージ（1/2）

コードから、システムの概要・目的を分析し提示可能

読み込ませたjavaファイルはどのようなシステムを構成するものか説明して

追加の命令

アップロードされた **192ファイルのJavaソースコード** をもとに分析した結果、このシステムは **地理情報システム**（GIS: Geographic Information System） に関連するものである可能性が高いと考えられます。以下の要素を含んでいることから、このシステムは **地図データの管理、解析、可視化を行うWebアプリケーション** であると推測できます。

どんなシステムか

---

1. システムの目的

- **地理空間データの管理・処理**
  地図データ、ベクターデータ、ラスターデータ（画像データ）を扱い、それらをデータベースに格納・管理する。
- **Webベースの地図可視化と操作**
  地図情報をWeb上で表示し、ユーザーが閲覧・操作できる機能を提供。
- **データ変換・解析機能**
  GISデータのフォーマット変換や、属性情報の抽出・解析を行う。
- **外部APIとの連携**
  地図サービス（WMS, WFS, WCS, SOS）などの標準プロトコルを利用して、外部の地理データと連携。

システムの目的

補足：検証パターン②の出力イメージ（2/2）

コードから、システムの構成・機能を分析し提示可能

2. システムの主な構成

このシステムは **3層アーキテクチャ（フロントエンド・アプリケーション層・データベース層）** を採用していると考えられます。

(1) フロントエンド（ユーザーインターフェース層）

- `MapOption.java`、`MapInfo.java` などのクラスから、地図情報を操作するUIが存在。
- `SVGRenderer.java`、`KMLRenderer.java` など、**SVGやKML形式の地図データを描画** する機能を搭載。
- `GeoJSONfileWrapperServlet.java` など、**GeoJSON形式のデータを扱うAPI** を提供。

(2) アプリケーション層（ビジネスロジック）

- `DBManager.java` により、データベースとのやり取りを管理。

システムの構成

3. 主な機能

以下のような主要機能が含まれると推測されます。

機能	対応するJavaクラス	説明
地図の表示	`MapInfo.java`, `SVGRenderer.java`	地図データの可視化を担当
レイヤー管理	`LayerInfo.java`, `LayerUpdateInfo.java`	地図上のレイヤー（例: 道路、建物、河川）を管理
地理データのインポート	`UploadShapeFileServlet.java`, `UploadRasterServlet.java`	Shapefileやラスターデータのアップロード処理

システムの機能

活用ポイント

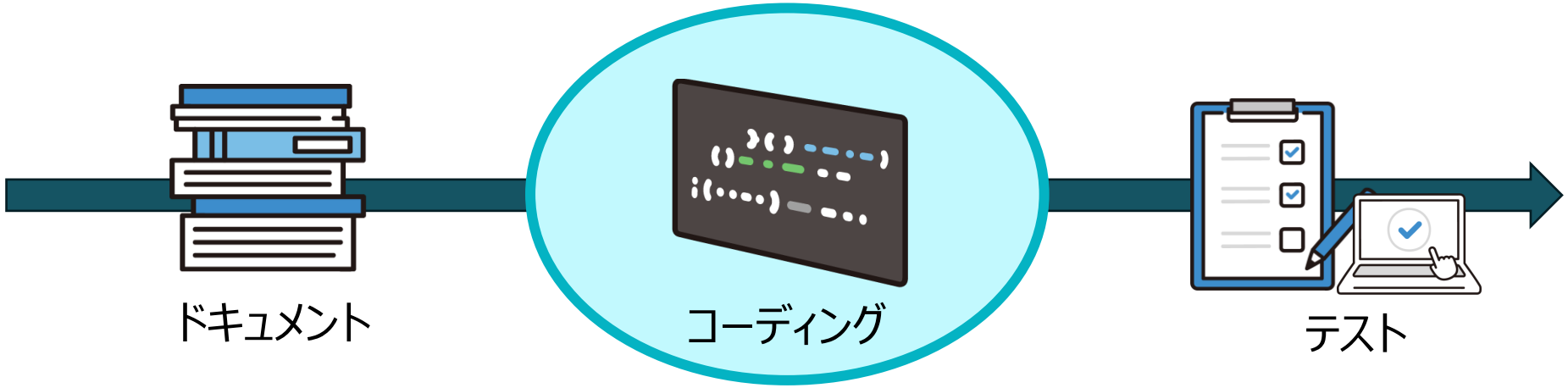
- 概要情報から設計書の**ベースの自動生成**、または**作成補助**が可能
- コードから設計文書への**リバースエンジニアリングを支援可能**

※成果物は、人による確認が必要

課題	課題解決の可能性	補足
コスト	△	文書作成、更新コストの削減に寄与 リバースエンジニアリングにおいては大きなコスト削減効果
組織・文化	○	ドキュメント作成のコスト減 ⇒ ドキュメントを残す文化を促進
統一性	○	プロンプトで書き方を指定 ⇒ 文書の構成の統一が可能

コーディング自動化

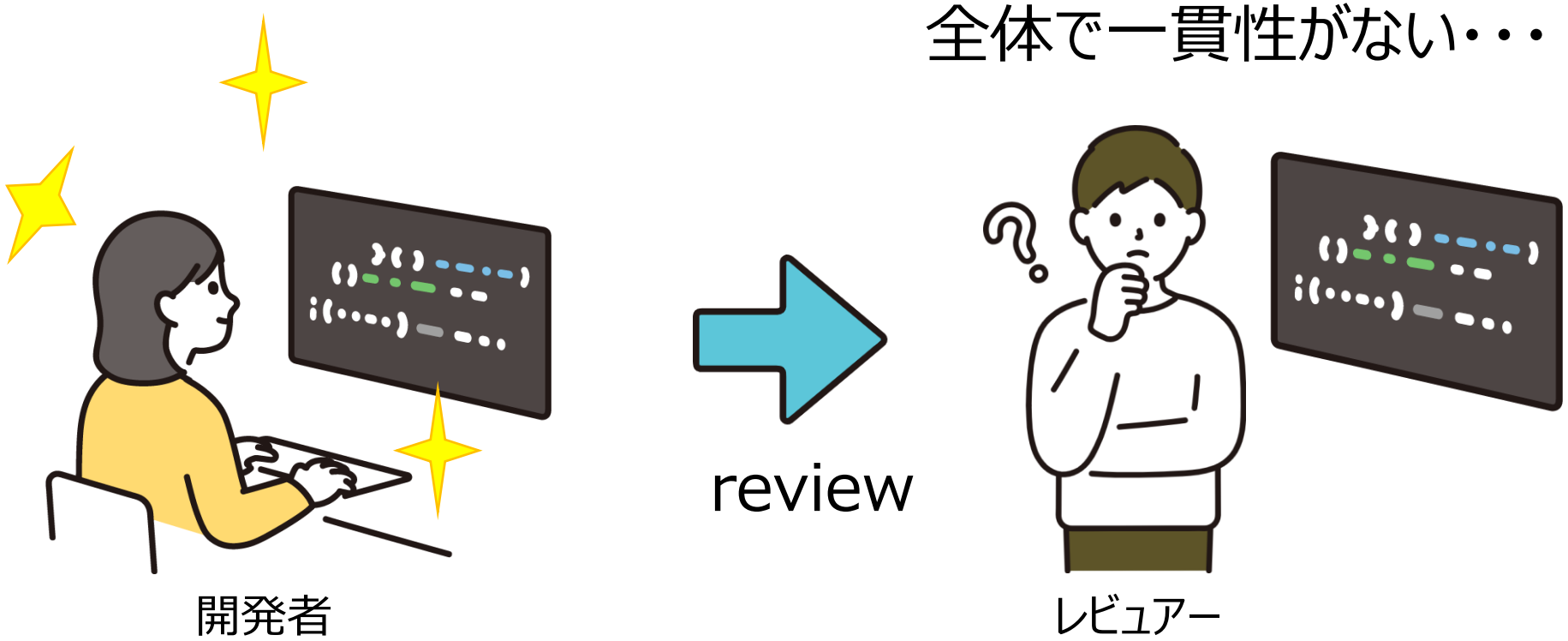
開発におけるコーディングの課題



課題　①「属人性」
②「生産性の向上」

課題①：属人化

理解に時間がかかる・・・
全体で一貫性がない・・・



保守性、開発効率 down

課題②：生産性の向上

自動化には達していない

ノーコード・ローコードツール

エディタの補完機能

検証方針

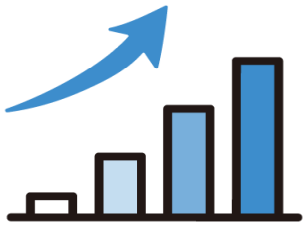
２つの課題を**AI**で解決する可能性を検証！

課題１：属人化

⇒ **誰でも一定品質のコードを作れるか？**

課題２：生産性の向上

⇒ **開発を効率化できるか？**



検証方法（1/2）

完成したプログラム／コーディング規約を模範として設定し、再現性を評価

模範

・【プログラム】航空券予約システム

・【規約】Google Java Style Guide

検証ツール

・GitHub Copilot

検証方法（2/2）

完成したプログラム／コーディング規約を模範として設定し、再現性を評価

1．航空券予約システムのコードから一部の機能を削除 

2．GitHub Copilotに削除した機能の作成を指示 

3．出力された結果を評価 

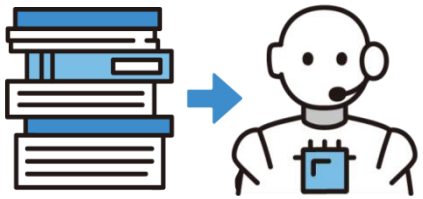
ポイント：コーディング規約に準拠しているか？
プログラム全体の整合性が保たれているか？

再現性が高い　⇒　規約に準拠した**属人化しない生産性の高い**コード

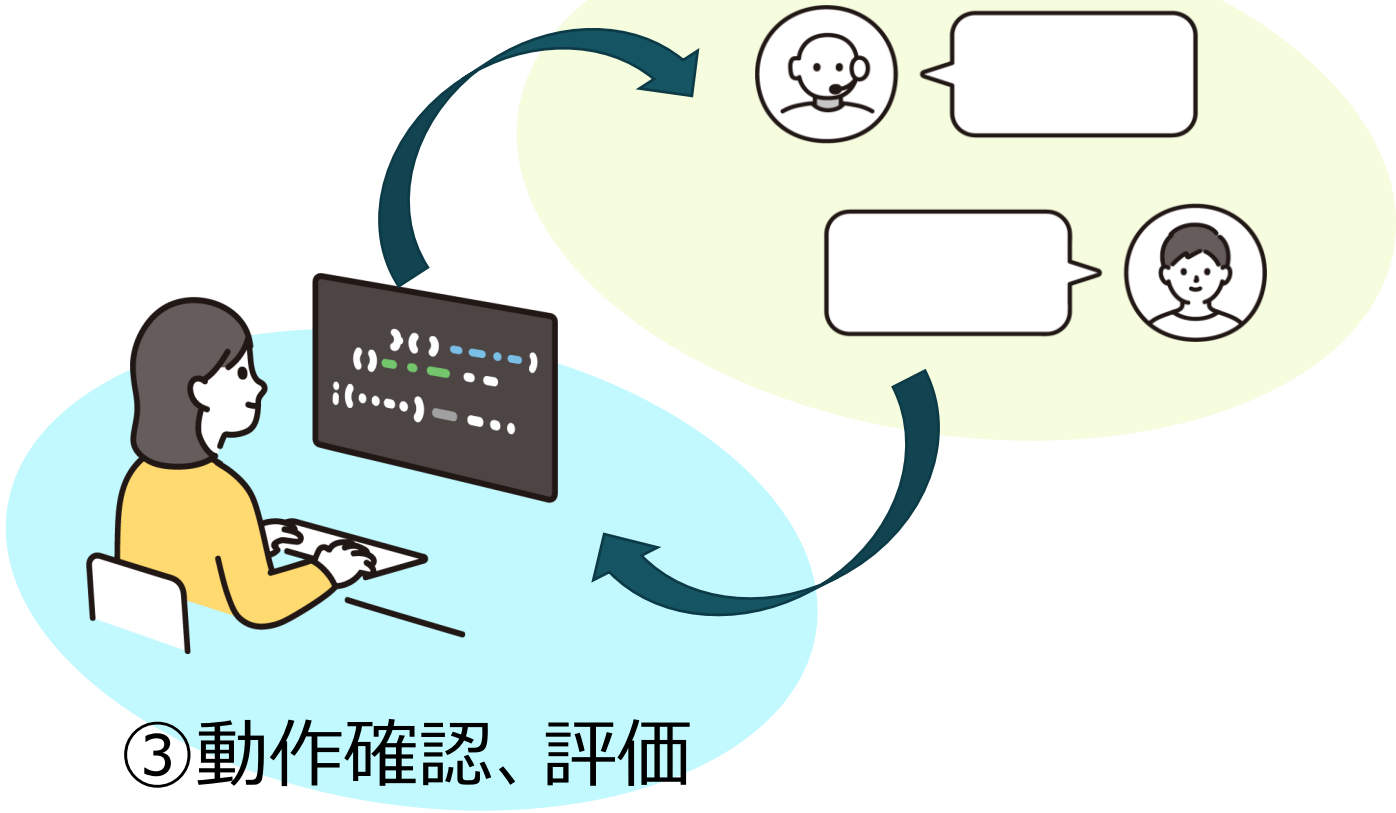
検証プロセス

いくつかの機能をピックアップして、再現性を評価

①AIに模範となるプログラム、
規約を与える



②コードの作成/修正を指示



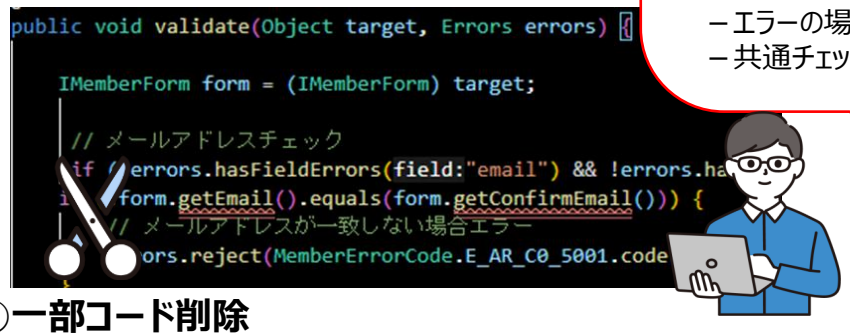
③動作確認、評価

検証結果：再現性〇

模範

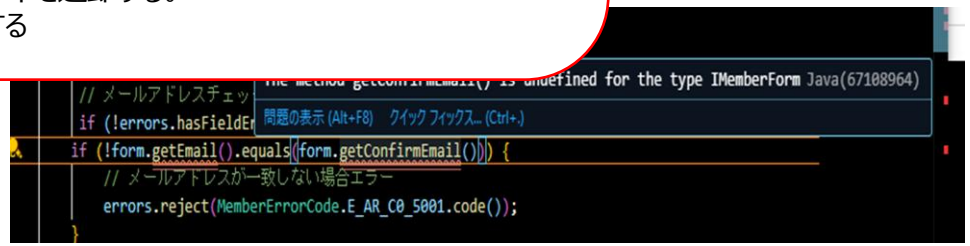
・Spring Frameworkを使って実装してください。
・インジェクトを明記してください。
・メソッドとして、以下の2つを実装してください。
　– バリデーターがサポートするかのクラスを判定するもの。
　– 実際にバリデーションを行うもの。
・「実際にバリデーションを行うメソッド」には、以下を含めてください。
　– バリデーション対象オブジェクトとエラーを保持するオブジェクトを引数とする。
　– キャストを使用する。
　– パスワードフィールドに既にエラーが無いかをチェック処理を入れる。
　– 「パスワード」と「再入力パスワード」が一致するかチェックし、一致しない場合はエラーを追加する。
　– エラーの場合は「E_AR_CA_5001」のエラーコードを返却する。
　– 共通チェックとして「invokeValidator」を使用する

指示



①一部コード削除

②再現指示



③コード生成＆提示

再現結果

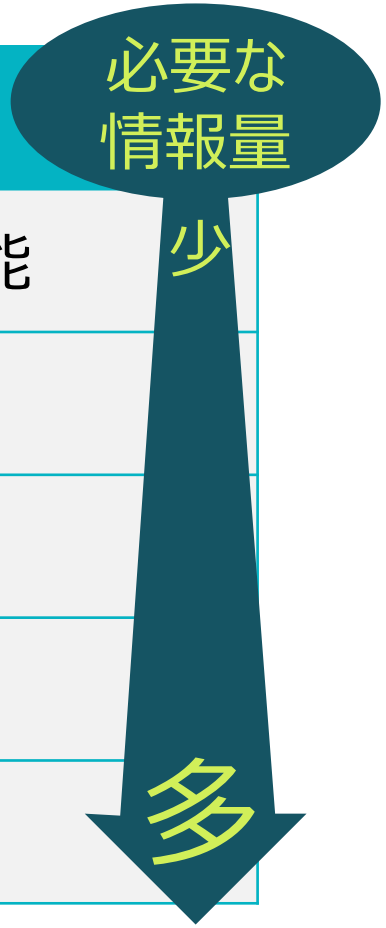
〇

必要な情報を指示することで、再現成功

検証結果

処理の内容と再現性

再現性	処理内容	補足
○	汎用的／一般的	日付処理やディレクトリ作成が可能
	仕様が明確	修正指示に基づき調整が可能
×	業務仕様が複雑	業務依存の処理は難しい
	DB構成に関連	複雑なDB構造の処理は難しい
	複雑なGUI	機能が不足しやすく調整が困難



考察

利用者のスキルとして明示的に指示すべきポイント

- フレームワーク
- 設計パターン
- 処理内容
- コードの一貫性

etc.

成功の実例

・Spring Frameworkを使って実装してください。
・インジェクトを明記してください。
・メソッドとして、以下の2つを実装してください。
  – バリデーターがサポートするかのクラスを判定するもの。
  – 実際にバリデーションを行うもの。
・「実際にバリデーションを行うメソッド」には、以下を含めてください。
  – バリデーション対象オブジェクトとエラーを保持するオブジェクトを
    引数とする。
  – キャストを使用する。
  – パスワードフィールドに既にエラーが無いかをチェック処理を入れる。
  – 「パスワード」と「再入力パスワード」が一致するかチェックし、
    一致しない場合はエラーを追加する。
  – エラーの場合は「E_AR_CA_5001」のエラーコードを返却する。
  – 共通チェックとして「invokeValidator」を使用する

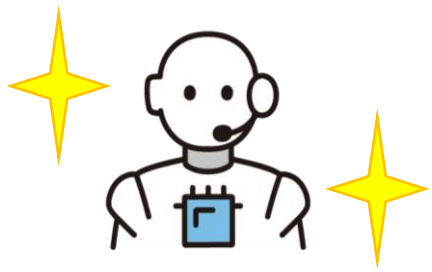
活用ポイント

①情報量　　：必要な情報を
②利用者のスキル：明確な指示で渡す

「情報量」　　「利用者のスキル」

＋

＋ etc. ＝



再現性
品質
スピード

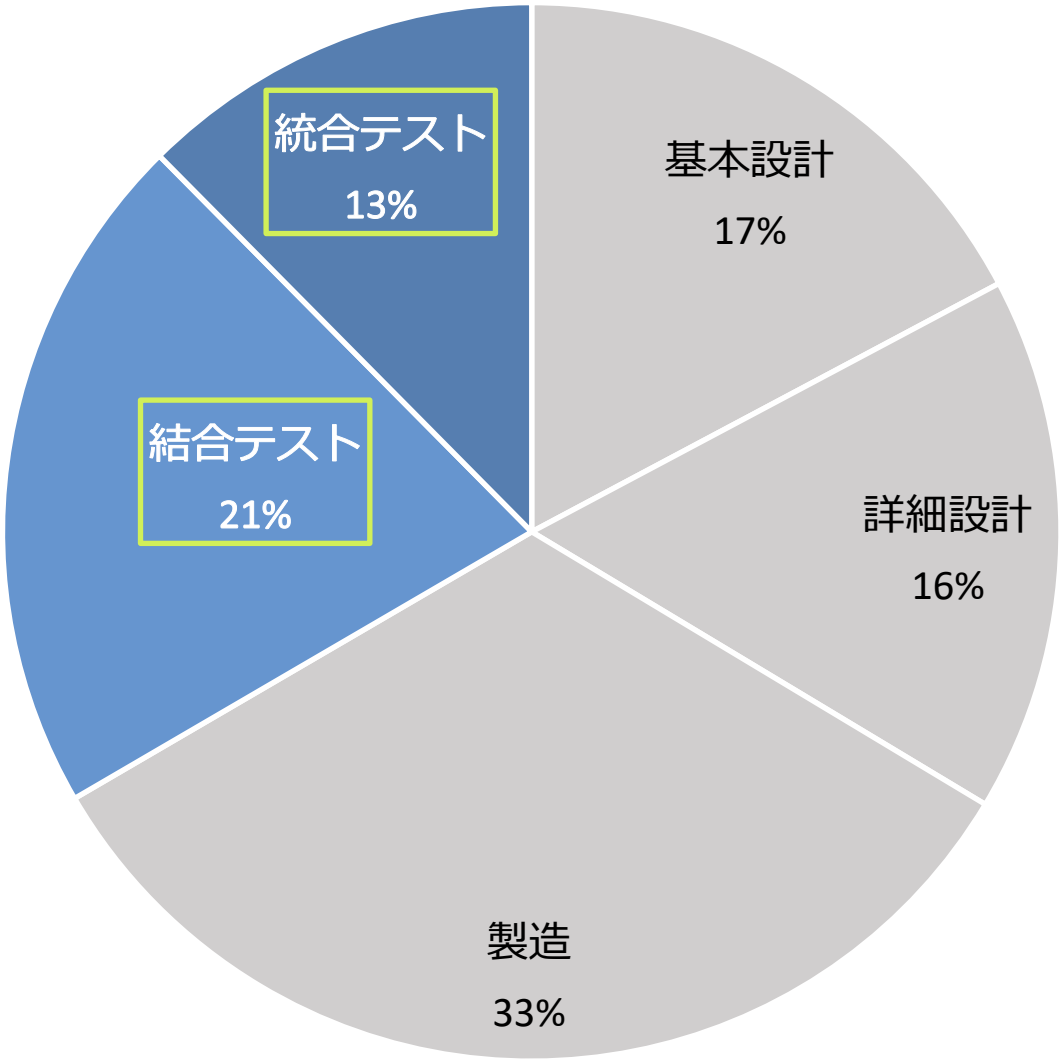
テスト設計自動化

テストの現状

システム開発において
テスト工程は**3割以上**を占める

テスト工程の削減・省力化は
システム開発全体のコスト削減につながる

新規開発における工程比率



『ソフトウェア開発分析データ集2022』
独立行政法人情報処理推進機構(IPA)を用いて作成

テストの課題

①期間の不足

- 前フェーズからの遅延の蓄積
- 変更できない納期

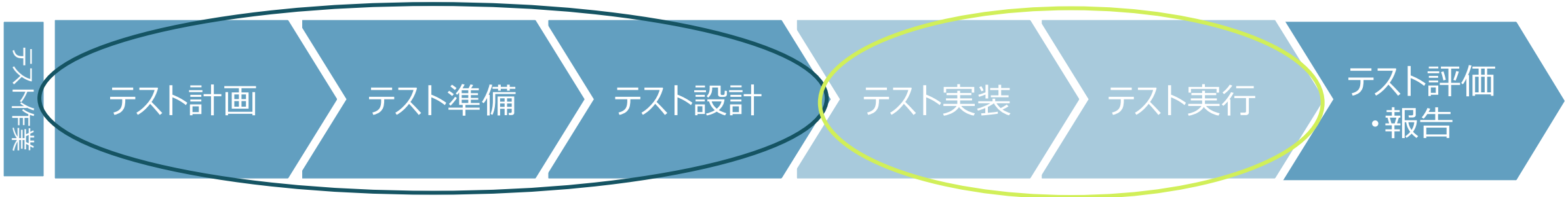
②人員の不足

- 仕様を理解した実施者の不足
- レビュー担当者の不足

③品質確保

- 手戻りのリスク
- 作成者による品質のばらつき

自動化・効率化の現状



効率化ツールはあるものの、
AIでの作成や判断は難しい…？

AIを用いた自動化が
進んでいる

- テンプレートや各作業向けのツールがあり、一部自動化や効率化が可能
- 方針の決定などには経験や知識に基づいた判断が必要となる

検証ターゲット

本検証では**テスト設計**をターゲットとしました

選定の理由

- テストケースの作成などに時間を要するため効率化が求められる
- 仕様や検証ルールに基づけば生成AIを利用して作成できるのでは？

検証の観点

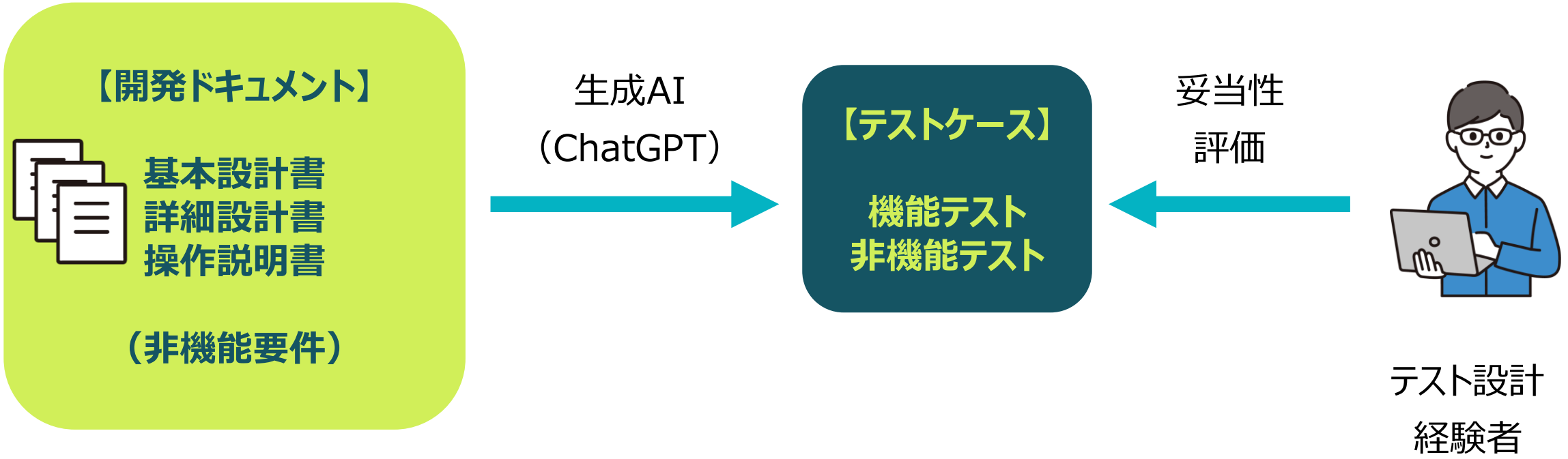
①正しいテストケースが作成できるか

- 各機能の仕様に対して間違っていないか

②抜け漏れなくケース作成ができるか

- 操作パターンや条件分岐に対応できるか
- 抜け漏れ確認のための判断ツールとなるか

検証方法



・開発ドキュメントの記載内容をもとに、生成AIでテストケースを作成

・テスト設計経験者がテストケースの妥当性を評価

生成AIでできたこと

○ ①正しいテストケースが作成できるか

- 各機能の仕様に対して間違っていないか

○ ②抜け漏れなくケース作成ができるか

- 操作パターンや条件分岐に対応できるか
- 抜け漏れ確認のための判断ツールとなるか

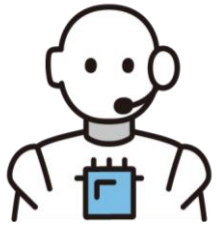
①ケース作成の効率化、②抜け漏れチェックの効率化が見込める

①テストケース作成の効率化

①インプット・指示

#役割 QA（品質保証）エンジニア
#前提事項 境界値、限界値テスト済
#指示事項
添付のファイルを読み込み結合テストケースを作成してください。
またケースを作成するのに必要な情報が足りていない場合は別途必要事項をまとめてください。
テストケースは表形式で抜け漏れチェックが行えるように基本設計書の項番を振ってください。

基本設計書・詳細設計書
操作マニュアル



追加の必要事項
以下の情報が不足しているため、詳細なテストケース作成にあたり追加情報が必要です：

ユーザー権限の詳細：…
エラーハンドリングの仕様：…
外部連携のテスト要件：…

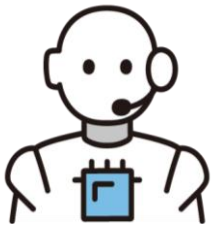
②不足情報の確認

③不足情報のインプット

不足情報については、以下を参照し、テストケースを作成してください。

#ユーザー権限の詳細
- xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

①テストケース作成の効率化



以下は、基本設計書の **3.1 eコミマップ作成機能** に基づいた結合テストケースです。

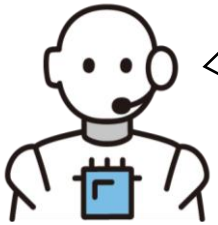
1. 結合テストケース（3.1.1 - 3.1.9）

項番	テストケース名	前提条件	操作手順	期待結果	実行結果	備考
3.1.1	マップ作成機能テスト	システム管理者権限があること	新しいマップを作成	マップが正常に作成され、データが登録される		基本設計書3.1.1参照
3.1.2	eコミマップ表示設定機能テスト	マップが作成済みであること	作成済みのマップの表示設定を編集	表示設定が正しく適用される		基本設計書3.1.2参照
3.1.3	項目設定機能テスト	作成済みのマップが存在すること	地図の表示項目を設定	項目設定が正しく適用され、地図に反映される		基本設計書3.1.3参照

②抜け漏れチェックの効率化

今回取り込んでいない基本設計書の項番と取り込まなかった理由を表形式で教えてください

以下は、今回の結合テストケースに取り込まれていない基本設計書の項番とその理由を表形式でまとめたものです。



取り込まれていない基本設計書項番一覧

項番	項目名	理由
3.1.5	クリアリングハウス登録機能	結合テストケースの範囲外であり、外部システム連携に関する機能のため、別途APIテストが必要です。
3.1.6	KMLレイヤ登録機能	KMLフォーマットに特化した登録機能であり、他のレイヤ登録機能との重複が多いため優先度が低いと判断されました。

考察

○ ①正しいテストケースが作成できるか
- 各機能の仕様に対して間違っていないか

○ ②抜け漏れなく
- 操作パターンや
- 抜け漏れ確認のための判断ツールとなるか

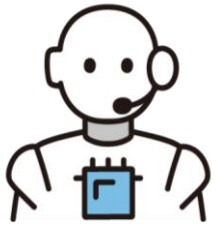
ただし、、、

テストケースの品質、検証方法の適切さ等は最終的に**「人間」**が判断する必要がある

生成AIにチェックリストを作成させることで、判断の省力化も可能

チェックリスト作成例

ここまで作成した項目が適切かを判断する方法はどのようなものがありますか？



1. 要件との網羅性の確認
2. リスクベースのテスト分析
3. ユーザビリティとユーザー操作のシナリオ確認

…

今あげてくれた方法をもとに、シナリオテストを作成する際に項目が適切か、抜け漏れがないかを確認するチェックリストを作りたいです。

カテゴリ	チェック項目	確認
要件網羅性チェック	システム要件に記載されているすべての機能に対して、少なくとも1つのテスト項目が存在していますか？	[ ]
	各機能の正常系シナリオが作成されていますか？	[ ]
	各機能の異常系シナリオ（エラーハンドリングや無効入力など）が含まれていますか？	[ ]
	重要な業務フローやユーザー操作のシナリオがシステム全体の流れに沿って網羅されていますか？	[ ]
リスクベースチェック	システムでリスクの高い機能に重点を置いたテストが含まれていますか？	[ ]
	エラーやデータ不整合が発生しやすい操作や、運用上のリスクがある操作に対する異常系テ	[ ]

活用ポイント

テスト設計のために並走するアシスタント

現時点での活用として以下のような用途が考えられる

- テストケースの初版作成
- ケース作成後の抜け漏れチェックに利用
- 自身での作業時になかった観点や気付きを得る

総論

現状のシステム開発自動化における結論

ドキュメント作成自動化

- 概要情報から設計書の**ベースの自動生**成、または**作成補助**が可能
- コードから設計文書への**リバースエンジニアリングを部分的に支援**可能

コーディング自動化

- **必要な情報**と共に**利用者が明確な指示を行う**ことで、**効率的なコーディング支援**が可能

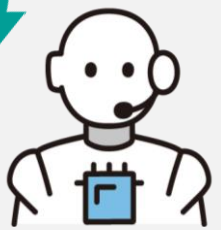
テスト設計自動化

- 基本的なテストケース作成、抜け漏れ確認のための**補助ツールとして活用**可能

システム開発自動化の未来

生成AIの進化に伴い、システム開発は人と生成AIとの分業が加速！

- **人はグランドデザイン・要件定義・システムアーキテクトに注力**
- **生成AIへのプロンプトエンジニアリングスキルは必須**

代表的な工程		役割
グランドデザイン	ー	 エンジニア  プロンプト 応答  生成AI
システムアーキテクト	要件定義	
	設計	
	実装	
	テスト	
	デプロイ	
運用と保守	ー	

ご清聴ありがとうございました