\Orchestrating a brighter world NEC

2023年度JEITA ソフトウェアエンジニアリング技術ワークショップ

NECの生成AIとソフトウェア・システム開発への取り組みについて

2024年2月9日

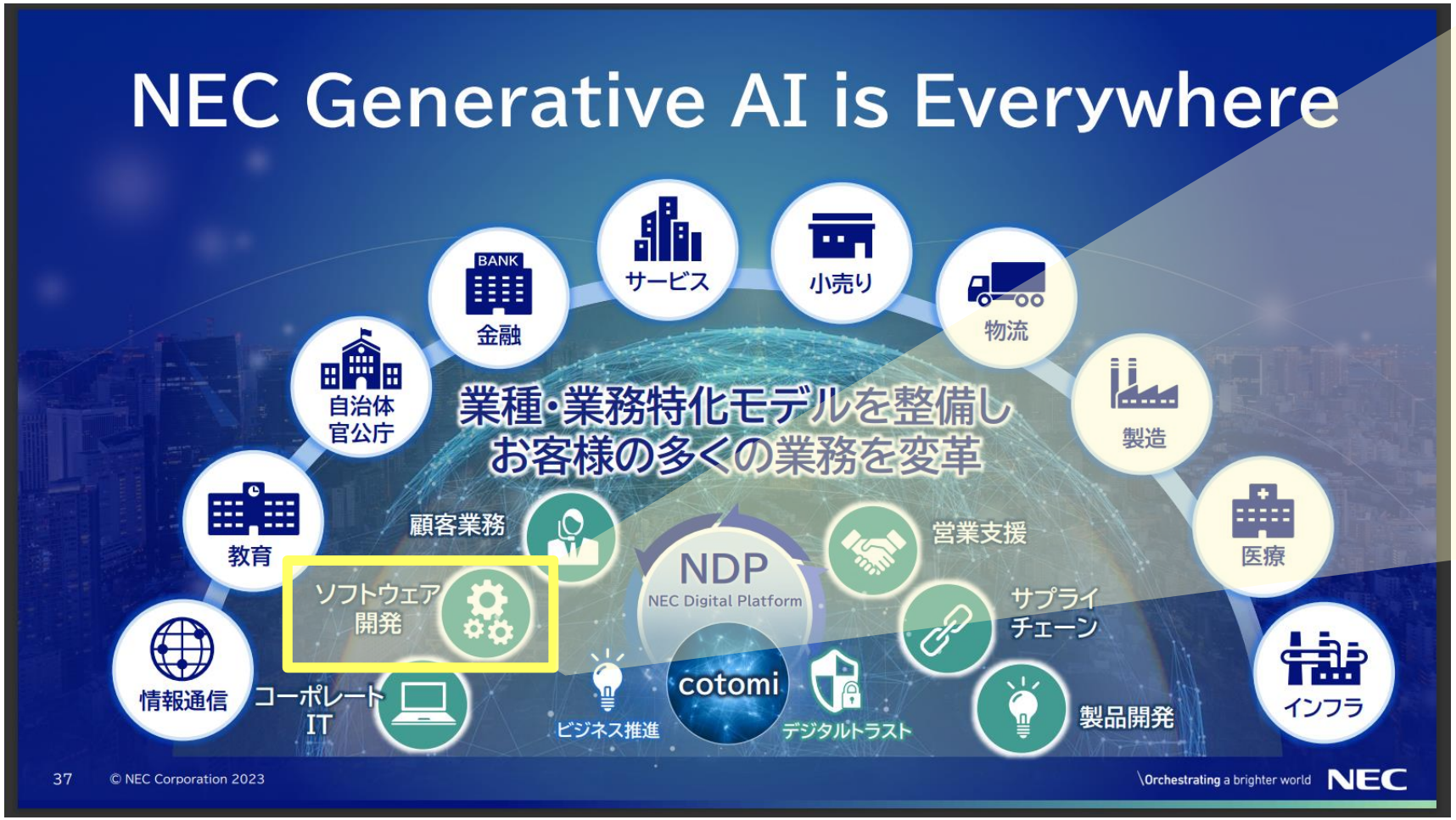
NEC　品質・エンジニアリング推進部門　ソフトウェア＆システムエンジニアリング統括部
矢野尾　一男

\Orchestrating a brighter world

NECは、安全・安心・公平・効率という社会価値を創造し、
誰もが人間性を十分に発揮できる持続可能な社会の実現を目指します。

はじめに

NECでの、生成AIのソフトウェア・システム開発への活用についてご紹介します



品質・エンジニアリング推進部門

ソフトウェアエンジニアリング
プロジェクトマネジメント
品質マネジメント

ソフトウェア・システム開発への
生成AI活用を推進

経営方針・事業説明会: IRイベント｜NEC

本講演の構成

以下の2部構成で進めさせていただきます

- ソフトウェア開発への生成AIの活用
  ソフトウェア・システム開発に生成AIがどのように使えるか

- NECグループでの取り組み
  NECグループでのソフトウェア・システム開発への生成AI活用の取り組みのご紹介

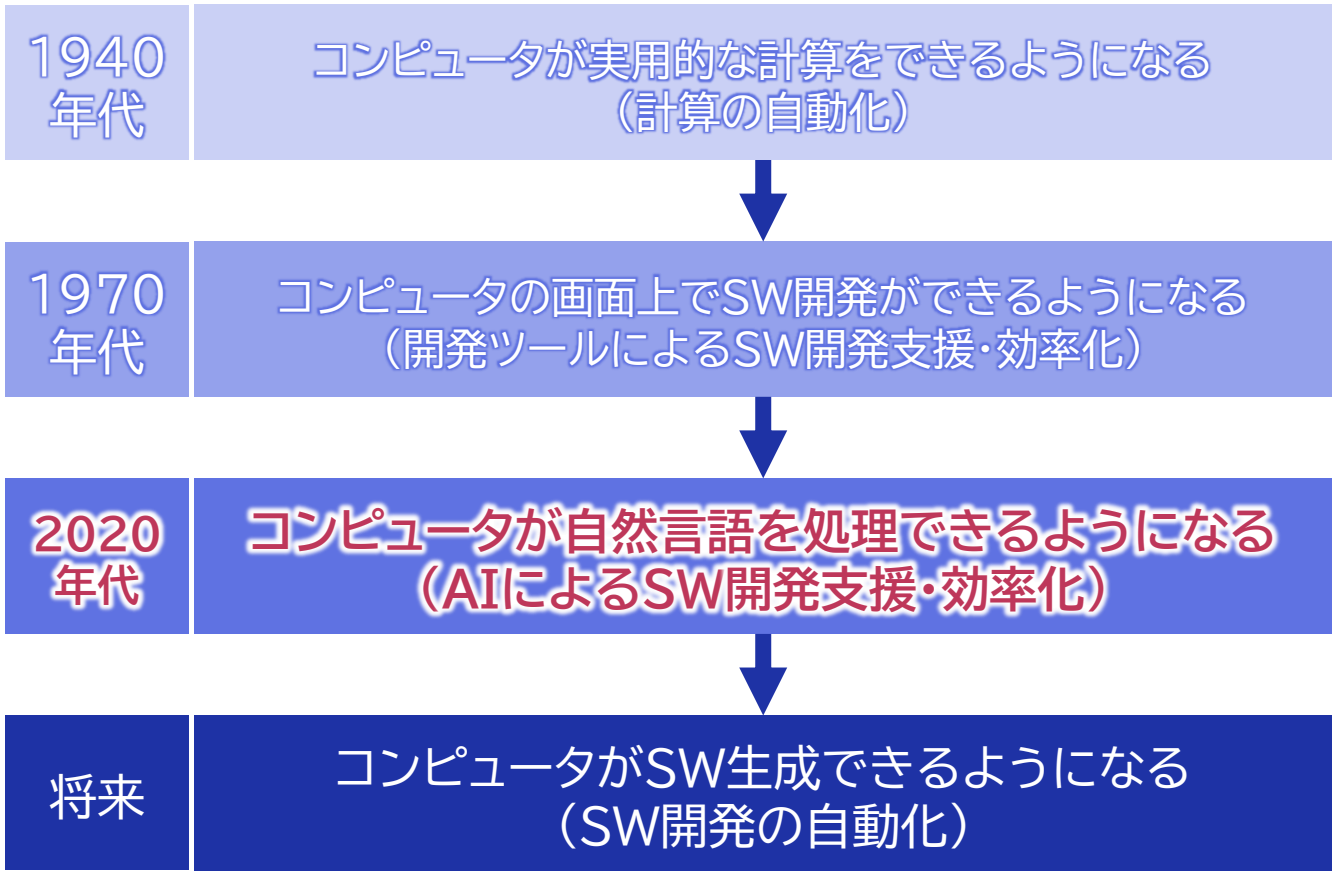
2. ソフトウェア開発への生成AIの活用

2.1 生成AIのソフトウェアエンジニアリングへのインパクト
2.2 開発プロジェクトで使う上での課題と対策
2.3 現時点での生成AI活用の指針

2.1 生成AIのソフトウェアエンジニアリングへのインパクト

自然言語で記述された要件や設計を、コンピュータが直接処理できるようになる
生成AIをどう活用するかが、今後のソフトウェア開発の最重要課題のひとつ

- IT人材不足などの課題の解消
- 品質・生産性の大幅改善



生成AIを活用しやすい例

Node.jsのマイクロサービスを生成AIを使って開発した例
モデルがよく知っている領域であり、小規模なので、様々なタスクで活用可能

フェーズ	用途	効果/課題
要件定義	要求の抜けの確認	適切な質問文を生成し、要件を補完できた
外部設計	アーキテクチャ設計	一般的なMSAを生成。認証処理をどうするかあいまいなまま進めてしまった
	バックエンドモジュール基本設計	一般的なマイクロサービスの設計、DB設計を生成。問題なし
	基本設計の抜けの確認	適切な質問を生成し、設計を補完
	機能仕様書(API仕様書)作成	一般的なREST APIの仕様を生成。問題なし
内部設計	内部設計書の作成	NodeJSの一般的なREST APIの設計を生成。問題なし
	実装手順作成	開発環境の設定などを生成。問題なし
コーディング	実装手順の各ステップの内容を作成	細かな誤回答が生じたが、誤りを指摘して修正させることができた ・要件との細かな不整合あり(入力要件よりも、モデル上の知識を優先している) ・生成されたコード中に、実装手順に書かれていないファイルのインポート処理があった ・認証処理の仕様変更に未対応のコードが生成された ・認証サーバーのURLをハードコーディングしていた ・BDレビューで修正した内容が反映されていない
	動作確認/デバッグ	実行はできるがエラー発生。エラーメッセージからはGPTは原因・調査方法を特定できず。 SQLログをGPTに入力したところ、修正方法を生成できた(1度間違えた)
単体テスト	テスト仕様・コードを生成	内部設計書とソースコードから、単体テスト仕様・コードを生成

生成AIを活用しやすい例

全自動でできるわけではなく、デバッグ完了までに37回の対話を実施

1回目：要件の入力と確認指示

プロジェクト管理システムの要件は以下のとおりです。要件に不明点が無いかを確認して、質問してください。

(要件)

4回目：要件定義書の生成指示

上記の内容をもとに、要件定義書を作成してください。

7回目：基本設計書の生成指示

まずバックエンドの実装を進めてください。バックエンドモジュールの基本設計書を作成してください。

13回目： 内部設計書の生成指示

上記の設計情報を元にして、バックエンドモジュールの内部設計書を作成してください。

14回目： 実装手順の生成指示

バックエンドモジュールの実装を開始します。実装の手順をstep by stepで教えてください。

21回目： ファイル単位のコード生成指示

middlewareディレクトリ内に作成する、authMiddleware.jsとauthorizationMiddleware.jsファイルの内容を教えてください

22回目：生成されたコードの不具合の指摘

上記のauthMiddleware.jsは、JWTを自分で発行して、自分が持っている秘密鍵で検証しているようです。今回の要件では、認証サーバーは別に存在しているので、JWTの検証は、認証サーバーに依頼する方式に変更してください。

34回目：実行エラーの指摘

作成したバックエンドを実行したところ、以下のようなエラーレスポンスが返ります。原因を教えてください。{"message": "Error retrieving projects:列"nan"は存在しません"}

35回目： 実行エラーログの入力

上記の問題はありませんでした。sequelizeのloggingを有効にしたところ、以下のようなログが出力されました。原因は分かりますか？
(略)

生成AIを活用しにくい例

ServiceNowの申請画面を生成AIを使って開発した例
ServiceNowの知識不足/旧仕様との混同のため、正しい手順を生成できなかった

タスク	用途	効果/課題
要求分析	要求の抜けの確認	適切な質問文を生成し、要件を補完できた
手順概要	設定手順の生成	ServiceNowの申請画面作成の設定手順を生成。一世代前のワークフローエンジンを使った手順が生成される
AP作成	アプリケーションの作成手順を生成	正しい手順を生成
カタログアイテム作成	カタログアイテム(申請画面)の作成手順を生成	実際の画面とは異なる手順が生成される。誤りを指摘しても正しい手順を生成させることができなかった。手順どおりの作業は不可
カタログ変数作成	カタログ変数(申請画面上の入力項目)の作成手順を生成	実際の画面とは異なり、ServiceNowの別の機能を混同したような手順が生成される。画面に何が表示されているかを知らせたところ、正しい手順を生成できた
	カタログ変数のバリデーション設定手順の生成	実際の画面とは異なる手順が生成される。誤りを指摘しても正しい手順を生成させることができなかった。手順どおりの作業は不可
フローの作成	Flow Designerでのフローの作成手順の生成	Flow Designerでの手順を生成するように指示したが、一世代前のワークフローエンジン (Workflow Editor)の手順が生成されてしまう。誤りを指摘すると、Flow DesignerとWorkflow Editorの手順を混ぜたような手順が生成されてしまう。手順どおりの作業は不可

生成AIを活用する際の課題

現実のプロジェクトで活用する場合、以下の点が課題となる
100%確実には動作しないので、現時点では人が結果を確認しなくてはならない

入力データ量制限

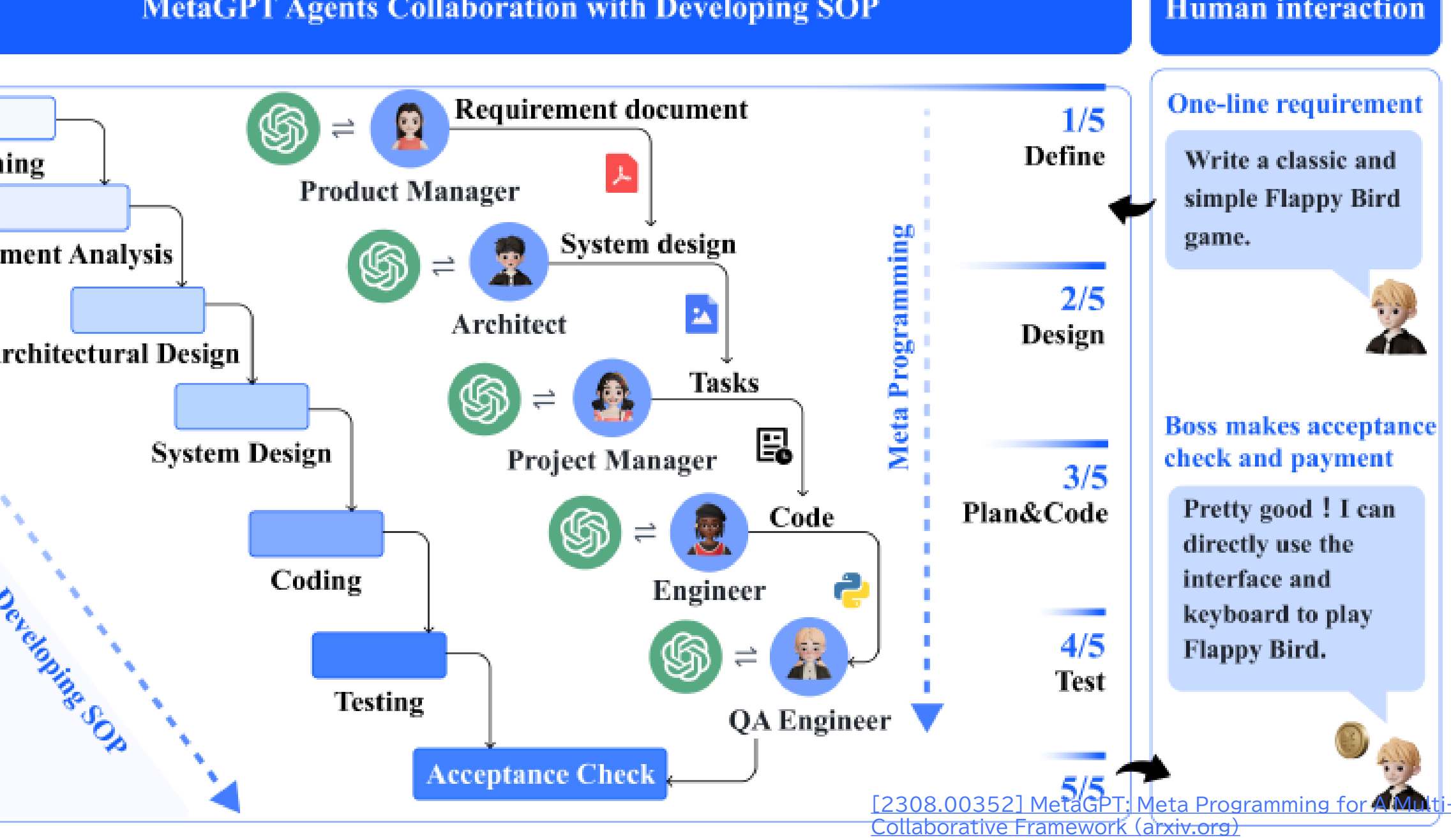
- 入力できる情報量(プロンプトの長さ)に限界がある。プロンプトが長くなると回答精度が落ちる

推論能力不足

- 次のトークンを予測しているだけであり、推論能力は高くない
- 確率的に動くので、いつも必ず正しい結果を返すとは限らない

専門知識不足

- 特殊なプラットフォームやプログラミング言語などに、直接には対応できない
- 業界/会社/プロジェクト特有の知識が必要なタスクに、直接には対応できない



[2308.00352] MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework (arxiv.org)

推論能力不足への対策

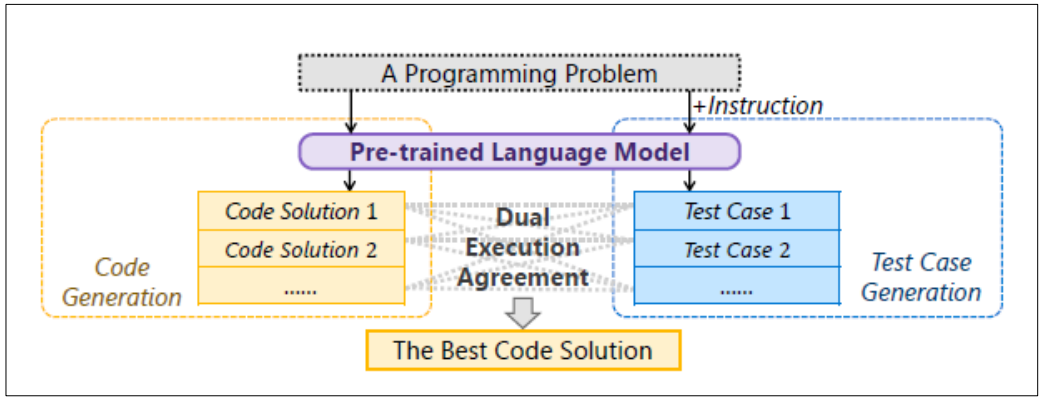
単純なタスクに分割する、計算処理などはLLMに任せず外付けする

回避策

- 単純(単体テストコードの生成、ひな型の生成など)に用いる
- 単純なタスクにかみ砕いて与える

低減策

- プロンプトエンジニアリング（CoTほか）
- タスクを自動分割する、外部プログラムと組み合わせる



CodeT: Code Generation with Generated Tests

単体テストも生成して、その実行結果を使ってどの候補を解とするかを判定

専門知識不足への対策

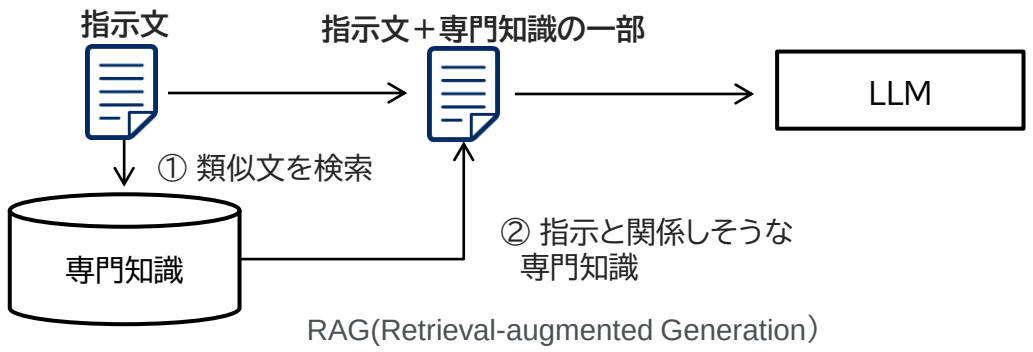
一般常識で推論できる領域で利用する、検索拡張/追加学習で対応する

回避策

- 多くのOSSで利用されている言語、プラットフォーム上の開発の下流工程で利用する

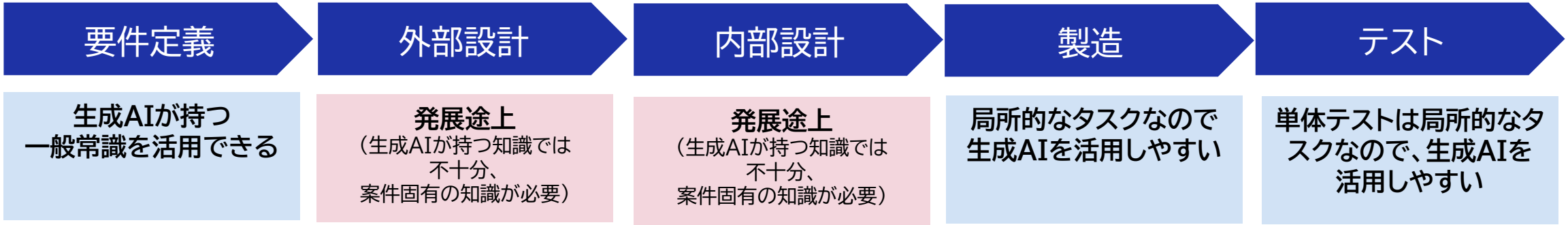
低減策

- 検索拡張(RAG)を用いる
- 追加学習で対応する。コストがかかるが、プロンプト長の問題がない



2.4 現時点での生成AI活用の指針

タスクの性質によって、活用のしやすさが異なる
案件固有の知識を検索/学習させることで活用領域を広げることができる



- 現時点では人による成果物の確認が必要なので、人の作業の補助として利用する
- 標準化することによって、高い効果（生産性/品質向上）が得られる
  - 活用のしやすさを評価して標準化
  - 検索しやすい設計情報の持ち方、生成AIに伝えやすい設計書などを規定
  - 共通のモデルを定義することで、RAG/学習の効率化・精度向上が見込める

3. NECグループでの取り組み

3.1 生成AI社内利用ルールの整備
3.2 生成AIを利用する開発ツールの提供
3.3 生成AIを前提とした開発プロセスの整備
3.4 上流工程の支援
3.5 品質管理支援
3.6 研究開発

3. 取り組み概要

プロジェクト支援・サポート

事例・プラクティス共有、COE機能

開発者向けエンジニアリング

ツール　3.2　生成AIを利用する開発ツールの整備

プロセス　3.3 生成AIを前提とした開発プロセスの整備

3.4 上流工程の支援

技術　3.6 研究開発

開発管理

3.5 品質管理支援

プロジェクト管理支援

リスク・コンプライアンス対応

3.1　生成AI社内利用ルールの整備

NECグループにおける生成 AI 社内利用の基本方針

積極的な利活用と安全性の両立を目的に、
秘密情報の取り扱いと、出力データの信憑性や権利侵害の課題に着目して制定

入力情報の秘密区分に従って適切に利用すること

生成AIへの入力情報秘密区分	生成AI利用可否（利用環境）
• 最高機密事項、極秘事項、秘密事項（個人情報を含む）	**利用不可**
• 秘密事項（個人情報を除く）、社外秘、NECグループ外秘	**利用可** • NEC Generative AI Service（NGS）＊ • NGS APIを利用する社内システム • 会社の許可した一部の外部SaaS
• 上記以外の情報	**利用可**

＊社内秘密情報漏洩対策として、データが再利用されず、個人認証、ログ保全などの機能を付加

以下のリスクを鑑み、出力コンテンツの品質チェックを実施の上、利用すること

正確性（信ぴょう性）に欠ける可能性がある

著作権・知的財産を侵害した内容である可能性がある

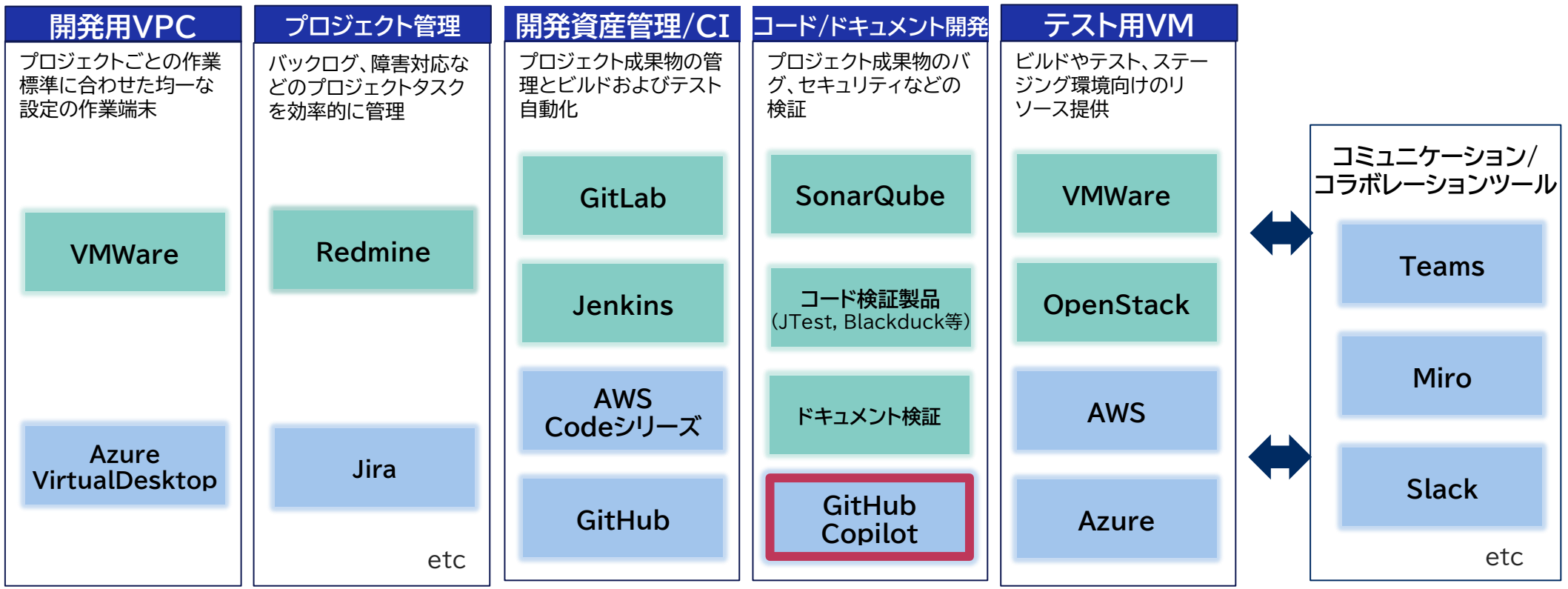
禁止事項

他者の著作権等の権利侵害につながる可能性があるプロンプトを入力すること

開発専用サービスの導入

ソフトウェアの開発環境をNECグループ向けに提供する「クラウド型SW開発基盤」で、GitHub Copilotの提供を開始

クラウド型SW開発基盤の主要サービス



3.2生成AIを利用する開発ツールの整備

開発専用サービスの導入

セキュリティリスクと著作権・知財侵害リスクへの対応のルールを規定
調達とサポートを一元化し、ガイド、事例、ベストプラクティスを提供/共有

セキュリティリスク管理

- GitHub環境のセキュリティ強化
- 生成AI利用基本方針との整合確認

著作権・知財侵害リスク管理

- 著作権保護機能の強制有効化。OSSリスク検査サービスとの併用を強く推奨
- 受託案件で利用する場合のチェック項目

利用ガイド・COE機能

- 既存開発プロセスでの利用ガイドを発行
- 事例・ベストプラクティスの共有

調達・サポート

- 調達とサポートを一元化

3.3 生成AIを前提とした開発プロセス

開発プロセスへの生成AIの取り込み方法を評価し、利用ガイドを発行

- 実装～単体テストへの生成AI利用ガイド
- 開発ガイドの生成AI対応
- モダナイゼーションガイド

実装～単体テストでの生成AI活用ガイド

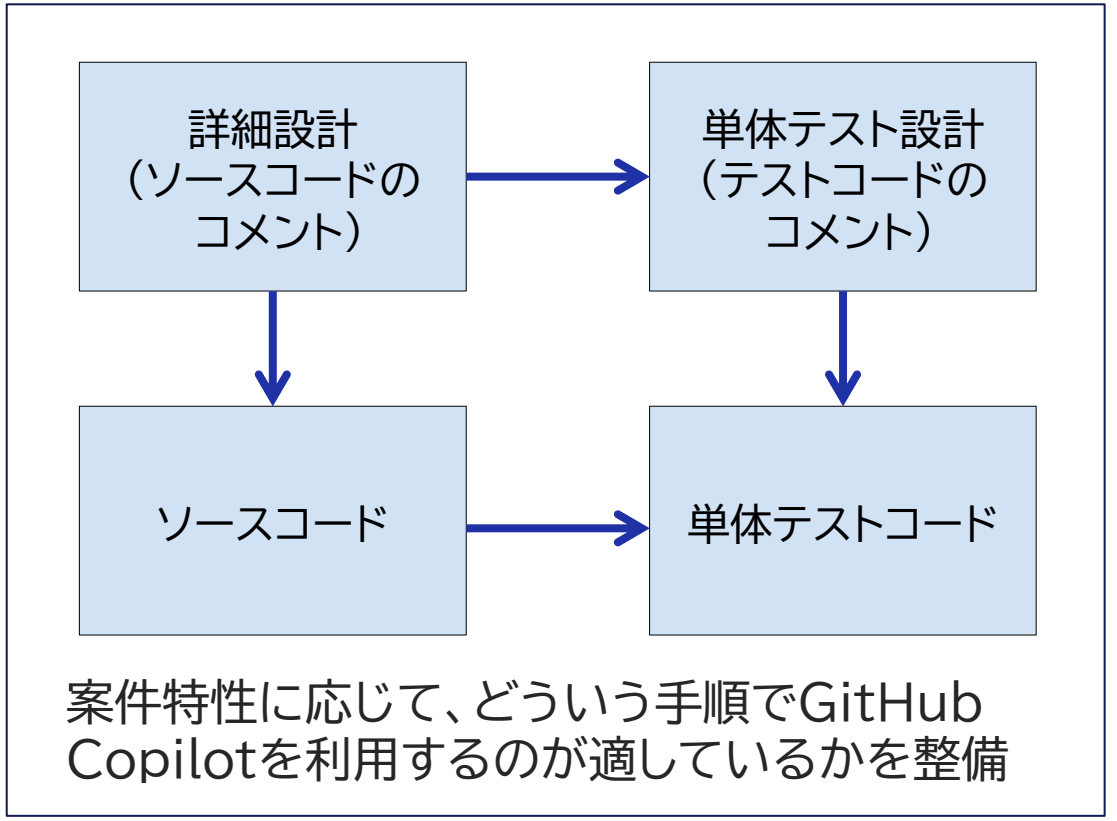
生成AI活用の効果が高い、実装～単体テストでの利用ガイドを作成して展開

実装～単体テストに注目する理由

- ◆生成AIを活用しやすい
  タスクが局所的なため、生成AIを活用しやすい
  GitHub Copilotを使って効率化しやすい

- ◆効果が大きい
  単体テストは、テストの中では一番量が多く工数がかかるため、効率化の効果が高い

- ◆状況に応じて活用方法が異なる
  使い方によって、品質にばらつきが生じる



案件特性に応じて、どういう手順でGitHub Copilotを利用するのが適しているかを整備

開発ガイドの生成AI対応

現行のSDE開発方法論のドキュメントに、生成AI編を追加
開発プロセスを変えるものではなく生成AIで効率化するという位置づけ(現時点では)

SDE: System Director Enterprise

- 開発プロジェクトの特性に応じた活用しやすさ
- 各タスクでの生成AIの活用しやすさ、実例
- 共通ノウハウ

課題	回避策	低減策
入力データ量制限	局所的なタスク(下流工程)で利用する 新規開発で利用する	タスクの分割 タスク分割の自動化
推論能力不足	単純なタスク(単体テストコード生成、ひな型生成など)に利用する	プロンプトエンジニアリング タスクの自動分割と外部呼出し
専門知識不足	多くのOSSで利用されている言語、プラットフォーム上の開発で利用する	検索拡張(RAG)または追加学習

モダナイゼーション

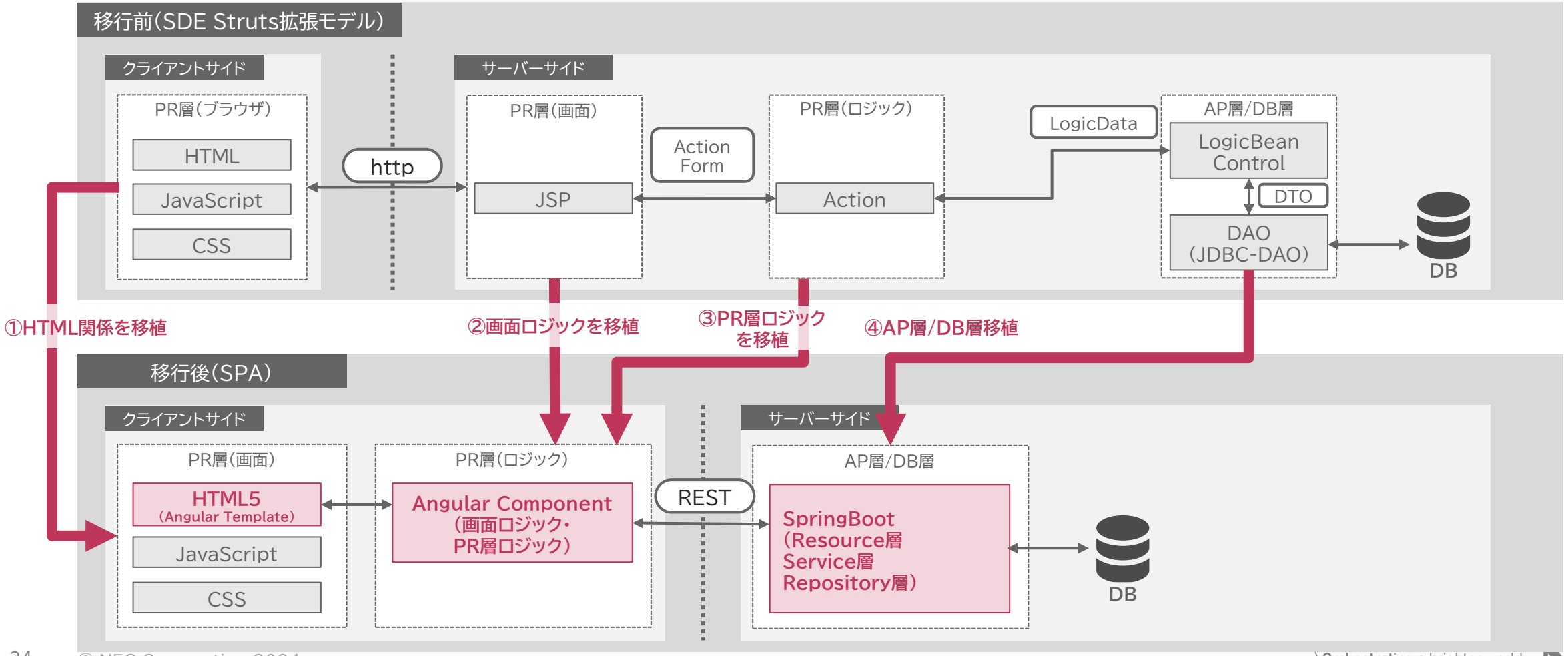
生成AI適用のニーズが高いため、モダナイゼーションへのプロセスを優先的に整備中

	類型が多い（モデル化SI※）	類型が少ない（スクラッチ型SI）
作り直し	• 標準的なモダナイプロセスを整備する • そのプロセスの各タスクのLLMの利用しやすさを評価してプロセスに組み込んでおく 標準モダナイプロセスへの生成AI組み込み	• 案件限りのモダナイプロセスをその場で作る • 事例を参考に、各タスクでLLMが利用できそうであれば利用して効率化する リバースエンジニアリング事例
移植	• 変換ツールを作り/調達し、モダナイプロセスを整備する。作成時に可能ならばLLMを利用する • 変換ツールでカバーしきれない箇所の人手タスクでLLMを使えるか評価してプロセスに組み込んでおく コンバージョンプロセスへの生成AI組み込み	• 案件限りの移植ツールを作るのはコストに見合わず、人手による移植になる場合が多い • 事例を参考に、移植作業をLLMで効率化できそうなら利用して効率化する コードコンバージョン事例

※SIノウハウを形式知化（テンプレート・ツール等）し組み合わせて行うSI

標準モダナイプロセスへの生成AI組み込み

既存のフレームワークのモダナイゼーションプロセスに生成AIの利用手順を組み込み



標準モダナイプロセスへの生成AI組み込み（続き）

各タスクにおける生成AIによる効率化の可否を評価し、標準ガイドに生成AI活用ガイドを追加

移行ステップ	タスク	適用可否	コメント
①HTML関係を移植	Angular Template/Component 生成	○	生成可能だが、修正が必要な部分あり
②画面ロジックを移植	validation 機能の移植	○	生成可能だが、修正が必要な部分あり
	メッセージリソースの移植		
③PR層ロジックを移植	Action クラス全体を一気に移植	×	完全な再現は不可能、内部の各種機能は個別実装が必要
	画面遷移	△～○	機能種別により、そのまま利用可能なコードを生成可能な場合と、生成が難しい場合あり どういった種別が活用可能か切り分け実施中
	API呼び出し	○	生成可能だが、修正が必要な部分あり
④AP層/DB層を移植	OAS の生成	○	ほぼそのまま利用可能なコードを生成可能
	APIサービス層の生成	検証中	
	他の機能の生成		

リバースエンジニアリング事例1

タスク分割と、検索拡張(RAG)機能を用いることで、アプリケーション画面とデータベースのテーブル間のCRUD図を生成

検証サマリ

環境	OpenAI のGPT-4環境を利用(RAG機能などを含む)
使用データ	WEBアプリケーションのSDE for Java※のサンプルソースを使用して検証を実施
課題	ソースコード全体を対象にしてCRUD図を生成することはできず、推論リンクが切れてしまうため、各階層(DB層・AP層・PR層)を辿って対応関係を抽出することはできない
実施方式	DB/AP/PR層の3層を個別に対応関係をLLMに推測させて、最終的に画面とテーブルの間のCRUD図を生成する。
検証結果	SDE for Javaのサンプルで、正しくCRUD表を生成することができることを確認

処理手順

STEP 1

DAO(DB層)の解析

DAOのソースファイルやMapperファイル、DB情報を、生成AIにアップロードし、DAOクラスとSQLとの関係情報を出力

STEP 2

AP層の解析

AP層のソースファイルと「DAOクラスとSQLとの関係情報」を生成AIにアップロードし、AP層クラスとDBテーブルのCRUD情報を出力

STEP 3

PR層の解析

PR層の画面ファイル、ソースファイルと「AP層クラスとDBテーブルのCRUD情報」を、生成AIにアップロードし、画面とDBテーブルのCRUD表を作成

※SDE for Javaについて:SystemDirector Enterprise for Java: 業務システム構築基盤 SystemDirector Enterprise | NEC

リバースエンジニアリング事例2

ソースコードや仕様書からの設計観点の生成

画面/API	設計観点		精度Lv	精度の説明	インプット情報	作成手順
画面	画面一覧		3	概ね安定して欠落無く出力出来ている	フォルダ構成 アプリケーション仕様	1. フォルダ構成を取得 2. NGSからプロンプトで出力指示
	画面遷移		2	ある程度出力出来ており内容もおおよそ合っている	画面一覧 ソースプログラム	1. 各画面それぞれに対して該当画面を基点にした画面遷移を出力 2. 1の情報を連結し、改めて画面遷移を出力
	個別画面仕様書	画面項目一覧	2	ある程度出力出来ており内容もおおよそ合っている	ソースプログラム	1. 各画面それぞれに対して表示/入力/ボタンの項目に分けて表形式で項目定義を出力 2. 1の情報をマージして項目定義を作成
		API呼出仕様	2	ある程度出力出来ており内容もおおよそ合っている	ソースプログラム	1. 各画面それぞれに対してAPI呼び出し一覧を出力
		アクション	2	ある程度出力出来ており内容もおおよそ合っている	ソースプログラム	1. 各画面それぞれに対してアクション一覧を出力
API						
	個別API仕様書	インターフェース仕様	3	概ね安定して欠落無く出力出来ている	ソースプログラム	1. 各ソースそれぞれに対して定義されているAPI毎のIN/OUTの情報を出力
		API処理仕様	2	処理対象のテーブル等に抜けがある	ソースプログラム	1. 各ソースそれぞれに対して定義されているAPI毎の処理フローを出力
		データ更新仕様	2	処理対象のテーブル等に抜けがある	ソースプログラム アプリケーション仕様	
		シーケンス図	2	処理対象のテーブル等に抜けがある	ソースプログラム アプリケーション仕様	

精度Lv	精度	説明
3	概ね信用して活用	ほぼ抜けや誤りは無いため、生産性向上が見込める
2	注意して活用	一部抜けがあるため、参考情報として活用する
1	信用できない	誤りがあるため、利用しない方が望ましい

前提：1KL程度のプログラムの場合の指標であり、プログラムの大小により差異が発生する

3.4 上流工程での支援

「作らないSI」で重要な領域。プロジェクト全体の情報を考慮する必要があるため検索拡張や追加学習を利用

- 検索拡張(RAG)サービスによる支援
- 独自LLMによる支援
- 専用ツールによる支援

検索拡張(RAG)サービスによる開発支援(検証中)

LLMフレームワーク(検索拡張(RAG)、タスクの自動分割、プロンプトの自動生成をサポート)上で、モデル化SI(※)の効率化を検証中

高スキル者の作業補助(生産性向上)

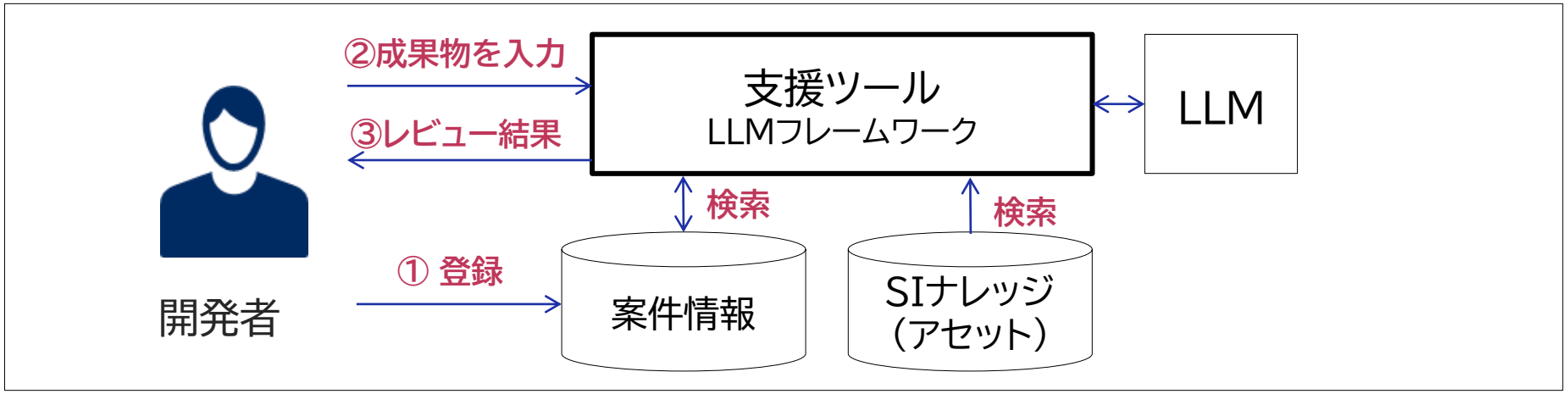
成果物の作成、他者の成果物のレビュー、各種付帯作業を生成AIで補助

生産性の向上により、スキルを必要とする業務により注力可能とする

低スキル人材の作業品質向上・教育支援

低スキル人材の作業を対話的に支援、成果物を生成AIで一次レビューすることで品質を底上げ

教育効果を高め、戦力となるまでの期間を短縮



※SIノウハウを形式知化(テンプレート・ツール等)し組み合わせて行うSI

3.4 上流工程での支援

独自LLMによるドメイン特化のドキュメントレビュー（検証中）

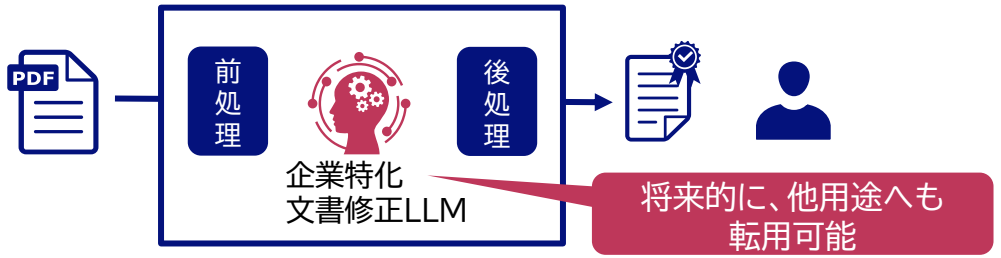
CS（誰の）	ドキュメントのレビューア
VP（どういう課題を解決するか）	ドキュメントチェックコストの削減
KPI・ 効果（削減時間、金額換算など）	ドキュメントチェックコスト・時間、ミス発見時の訂正工数
汎用LLMと比べた特長	・過去事例を学習することで、その企業独自の用語やチェック観点を考慮した回答を生成することができる ・日本語能力が高いので、日本語のドキュメントに対する回答精度が高い ・イントラネット内に構築できるのでセキュア

現状（課題）

- ドキュメントに対して、目検で注釈や数字の誤りに不備がないかを確認しており、工数がかかる。
- 各種チェックは複数名で点検するが、誤字・脱字や用語のケアレスミスを完全になくすことが難しく、また、精度が担当者スキルに依存している。

今後（LLM適用後のイメージ）

- 前処理の自動化と、文章チェックを一気通貫で行うシステムを用いて、均質的な文章チェックが可能
- 前処理の自動化も含み、チェック者の工数が削減される
- 過去の豊富な過去データも利用したLLMを作成でき、他の文章チェック業務へも応用可能



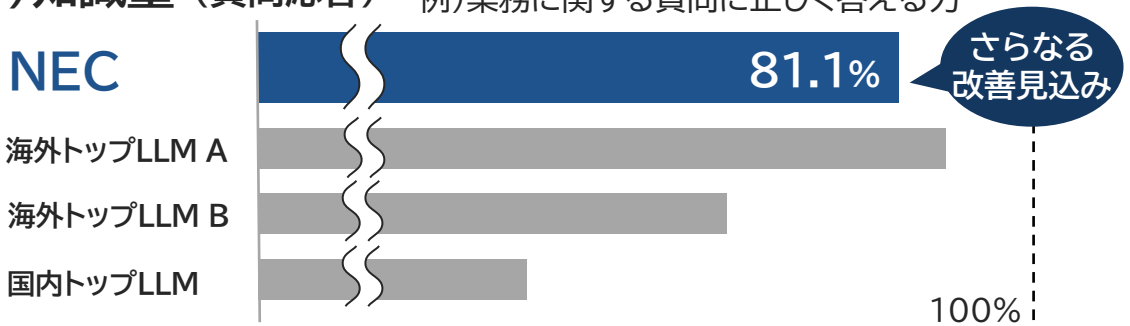
補足：NECでのLLMの開発

高い日本語能力と軽量さを両立したNEC開発のLLMを提供可能

特長① 高い日本語能力

英語/プログラミング言語に加えて、日本語でも優れた精度

(1)知識量（質問応答） 例)業務に関する質問に正しく答える力



(2)文書読解力（推論能力） 例)複雑で長い文章を要約する力

JGLUEベンチマークの JCommonsenseQA, JSQuADスコア（当社調べ）

特長② 軽量

1/13のモデルサイズの実現により、コストパフォーマンス向上

軽量化がもたらすお客様のメリット

- 運用時のサーバコスト・消費電力を抑制
- 業務アプリケーションでのレスポンスが高速
- お客様向けにカスタマイズしたLLMを短期間で作成
- オンプレ化が可能 秘匿性の高い業務でも利用可

3.4 上流工程での支援

専用ツールによる要件定義支援(検証中)

設計の事例を学習したLLMを利用して、要件からの設計パラメータの生成を支援
モデル化SI(※)の効率化を検証中

課題認識

2025年にIT技術者43万人不足※

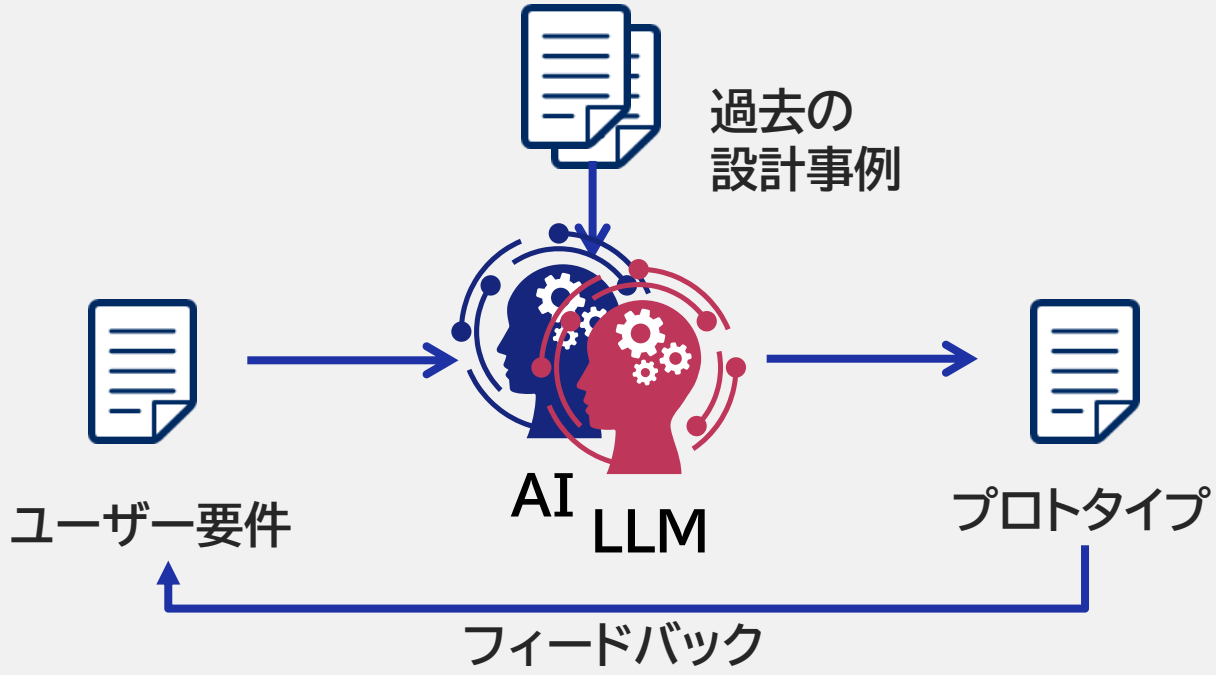
- 先端IT人材の供給不足
- 2025年までの
  ITシステム刷新が急務

膨大なSW、サービスの組合せ

- 例:現在でも多種・大量のOSS
  (約500種)

※ 経済産業省「DXレポート~ITシステム「2025年の崖」の克服とDXの本格的な展開~」

AIを活用した要件定義支援



※SIノウハウを形式知化(テンプレート・ツール等)し組み合わせて行うSI

3.5 品質管理での活用

SW開発における品質管理のタスクで生成AIを活用

- ソフトウェアのバグ分析（ODC分析）の支援
- なぜなぜ分析の支援

3.5 品質管理での活用

生成AIを活用したODC分析による評価項目の導出

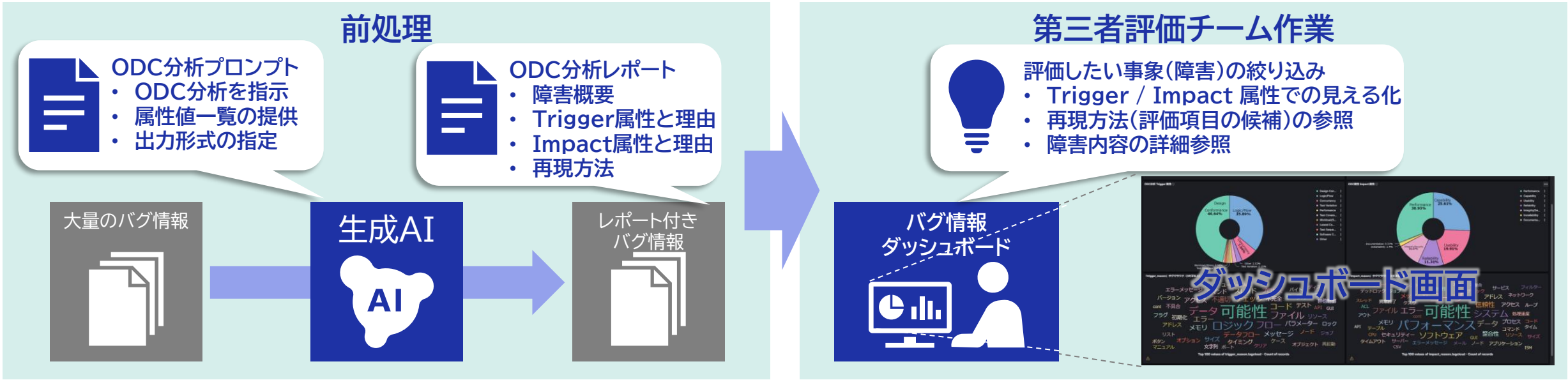
生成AIを活用して、フィールドバグ情報を入力に障害内容を要約すると共に、不具合が表面化した原因とその再現方法(評価項目の候補)を生成する

障害内容を**ODC分析の評価者視点(Trigger属性とImpact属性)で分類(属人性の排除)**

大量データを有識者だけで分類する限界、および大勢による分類では排除できない個人的な主観を解決

第三者評価チームでの**評価項目の作成作業を効率化(経験の浅い人材の底上げ)**

Trigger 属性と Impact 属性を参考に障害内容を絞り込み、再現方法(評価項目の候補)に容易にたどり着ける



3.5 品質管理での活用

生成AIを活用したサービス障害に対するなぜなぜ分析

生成AIを活用したなぜなぜ分析ツールを作成
障害発生情報をもとにしたなぜなぜ分析と対策立案をAIが支援

障害情報

実施時間	会社名G名	名前	実施内容	問題点
6:08	AA社	Oさん	ランプチェックシート、物理鍵をネックストラップにかけ対象ラック(ES-3)着 その他質問) *?ランプチェックシート=手順書(キングファイル)でよいか？* *→ チェックシートと手順書は別* *?キングファイルの重さは？ → 2kg程度* *?キングファイルは手に抱えて移動？ → キャリーバッグに入れて移動* *?キングファイルはワゴン等に載せてい移動？ → ワゴンは利用していない 媒体交換時はワゴンを利用し移動している*	(1)キングファイルを落とした理由 →早く終わらせようとした (2)キングファイルは重たい？ →重くはないがラック開施錠時のキングファイルの持ち方に問題がある (3)眠い？ →朝の3時まで休憩は取っていた 休憩時間は適切に取っている
6:08	AA社	Oさん	ラック前面の扉、物理鍵で開錠。 その他質問) *?鍵は鍵穴にさしたまま？ → 開けたら鍵は抜く*	
6:08	AA社	Oさん	ランプチェック開始。 その他質問) *?1人で実施、2人で実施？ → 1人*	
6:15	AA社	Oさん	対象ラック(ES-3)ランプチェック完了	
6:15	AA社	Oさん	ラック前面扉、施錠のため鍵穴に物理鍵を挿す	
6:15	AA社	Oさん	物理鍵を挿した状態で、ランプチェックシート(キングファイル)を落とす	
6:15	AA社	Oさん	ランプチェックシート(キングファイル)を拾おうとし、しゃがみ込む	
6:15	AA社	Oさん	しゃがみ込んだとき、ネックストラップに物鍵が引っ張られ、物理鍵中折れ発生	(1)鍵が老朽化していたから折れた可能性がある (2)ストラップは伸縮性のないものだった
6:17	AA社	Oさん	現場からPHSでエスカレーション実施(Oさん→アシスト シフトリーダー Wさん)	
6:25	AA社	Oさん+Wさん	対象ラック(ES-3)に到着	
7:25	AA社	Oさん+Wさん	2人で物理鍵中折れを確認	
7:36	AA社	Wさん	OP執務室へ戻り、上位へのエスカレーション実施するが、相手不出	
8:40	AA社	Hさん	着信履歴に気が付き、折り返し、OP執務室へcallし インシデント内容を把握。マニュアルに沿った、インシデント対応を指示(Hさん→OP Aさん)	
8:46	AA社	Wさん	再度、BB社 AOさんへエスカレーション(電話)。 AOさんから現地状況整理(事象・ラック製造番号確認等)を指示	
8:55	BB社	Oさん	BB社 内でのエスカレーション実施するが、相手不出	
9:05	BB社	Oさん	電話で追加指示OさんからWさん。ラックメーカー確認。	

生成AI 分析結果

【分析結果】
① 直接埋め込み原因:鍵の老朽化
② 直接摘出モレ原因:鍵の状態を確認しなかった
③ Mshellの観点からの間接原因:関係者の観点 - Oさんの不注意、ソフトウェアの観点 - 鍵の管理方法、ハードウェアの観点 - 鍵の耐久性、環境の観点 - 鍵の使用環境
④ 根本埋め込み原因:鍵の管理・メンテナンス体制の不備
⑤ 根本摘出モレ原因:定期的な鍵の点検・交換が行われていない

【判断根拠】
① 直接埋め込み原因根拠:(1)鍵が老朽化していたから折れた可能性がある
② 直接摘出モレ原因根拠:事実確認シートに鍵の状態を確認する記載がない
③ Mshellの観点からの間接原因根拠:(2)ストラップは伸縮性のないものだった、(3)眠い？→朝の3時まで休憩は取っていた
④ 根本埋め込み原因根拠:事実確認シートに鍵の管理・メンテナンスに関する記載がない
⑤ 根本摘出モレ原因根拠:事実確認シートに定期的な鍵の点検・交換に関する記載がない

【備考】不足情報:鍵の管理・メンテナンス体制、定期的な鍵の点検・交換の有無

【原因の深堀】
① なぜ1:なぜ鍵が老朽化していたのか？
⇒鍵の管理・メンテナンスが不十分だったため
② なぜ2:なぜ鍵の管理・メンテナンスが不十分だったのか？
⇒適切な管理・メンテナンス体制が確立されていなかったため
③なぜ3:なぜ適切な管理・メンテナンス体制が確立されていなかったのか？
⇒組織内で鍵の管理・メンテナンスに対する意識が低かったため

【根本原因に対する対策】
①根本埋め込み原因への対策
1. 鍵の管理・メンテナンス体制の見直し
2. 鍵の定期的な点検・交換の実施
3. 関係者への鍵管理・メンテナンスに関する研修・啓発活動

②根本摘出モレ原因への対策
1. 鍵の状態確認を含む作業手順の見直し
2. 鍵の状態確認を行う担当者の設定
3. 鍵の状態確認の重要性を関係者に周知する

3.6 研究開発

ソフトウェア・システム開発の領域へのLLM活用の研究開発の取り組み

- LLMによるセキュリティリスク自動診断
- データ分析の自動高速化技術

LLMを活用したソフトウェア・システム開発・運用支援技術

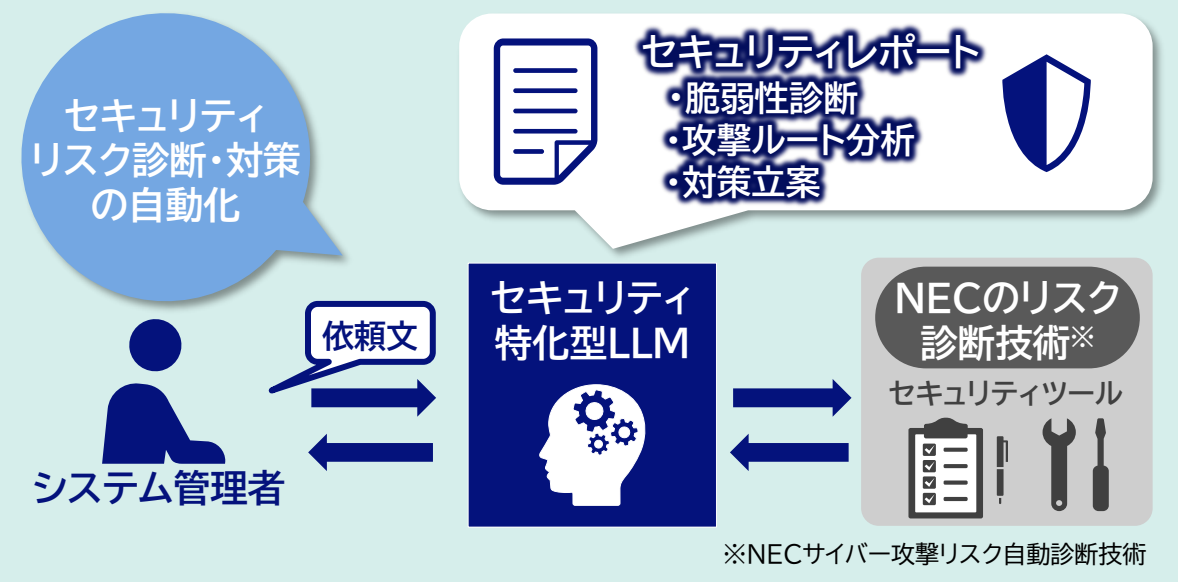
セキュリティ×LLM

LLMによるセキュリティリスク自動診断

R&D

- 非専門家でも、セキュリティリスクの診断が可能に
- セキュリティ特化型LLMがNECリスク診断技術を活用して、依頼文に対し診断やレポートを短時間に回答

例：脆弱性診断、攻撃ルート分析、対策立案



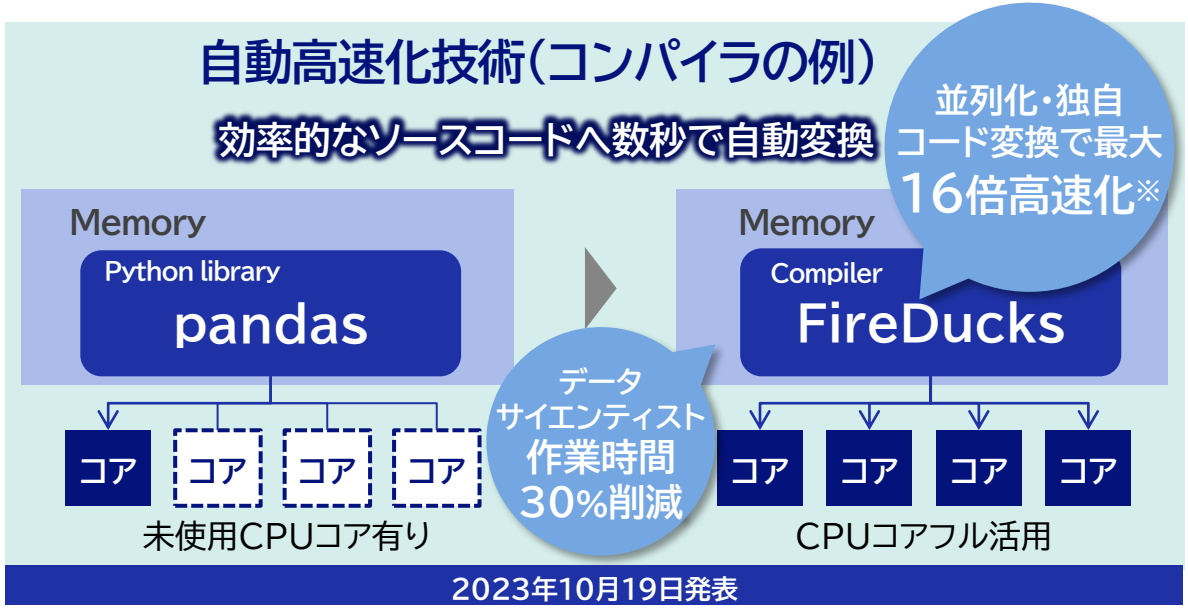
プログラムの自動高速化

データ分析の自動高速化技術（コンパイラ、アクセラレータ）

R&D

- 負荷の高いテーブル処理を16倍、Deep Learning処理を4倍高速化。さらにユーザプログラムをLLMで高効率化
- データサイエンティストの作業時間30%削減に相当

※β版公開中 https://fireducks-dev.github.io/



4.まとめと今後の展望

4.1 まとめ

◆ ソフトウェア開発への生成AIの活用

- 現時点では人の作業の補助という位置づけ
- 適不適があるため、どのタスクでどう使うかの検証が必要
- 開発プロセスの標準化が今まで以上に重要になる

◆ NECグループでの取り組み

- これまで培ってきたソフトウェア・システム開発のアセットと融合した取り組み
- 下流工程だけでなく、システム開発に関わるあらゆる領域が対象
- 標準開発環境にアセットを集約し、NECグループ全体に展開
- セキュリティや著作権侵害等、生成AI活用におけるリスク等についても対策を整理し、正しい使い方を確立すること

4.2 今後の展望

- 人の補助ではなく、人とAIが共同で開発を進めるようになる日も遠くなさそう
- SIベンダー企業が提供する価値は今まで以上に上流にシフト。DXレポートで指摘されているビジネスモデルの転換が加速される

デジタル産業の企業類型

① **企業の変革を共に推進するパートナー**	• 新たなビジネス・モデルを顧客とともに形成 • DXの実践により得られた企業変革に必要な知見や技術の共有 • レガシー刷新を含めたDXに向けた変革の支援
② **DXに必要な技術を提供するパートナー**	• トップノッチ技術者（最先端のIT 技術など、特定ドメインに深い経験・ノウハウ・技術を有する）の供給 • デジタルの方向性、DXの専門家として、技術や外部リソースの組合せの提案
③ **共通プラットフォームの提供主体**	• 中小企業を含めた業界ごとの協調領域を担う共通プラットフォームのサービス化 • 高度なIT 技術（システムの構築技術・構築プロセス）や人材を核にしたサービス化・エコシステム形成
④ **新ビジネス・サービスの提供主体**	• IT の強みを核としつつ、新ビジネス・サービスの提供を通して社会への新たな価値提供を行う主体

出展：DXレポート2.1

生成AIの進化により、ビジネスモデルの転換が加速される

業種・業務に関するドメイン知識を持つ専用LLMの強化がより重要になる


Orchestrating a brighter world
NEC