失敗から学んだノウハウを有効活用できる組織にするには

～ ソフトウェア品質向上のための生成AI活用法の考察 ～

グループ５＆６
上田 浩 （エンカレッジ・テクノロジ株式会社）
栗原 純一 （株式会社日立ソリューションズ・クリエイト）
黒柳 元 （株式会社アイシン）
津久井 秀樹 （キヤノンメディカルシステムズ株式会社）[発表者]
永田 哲 （元キヤノン株式会社）
深水 廉子 （SCSK株式会社）
宗田 貴之 （永山コンピューターサービス株式会社）
横尾 清吉 （株式会社富士通ゼネラル）
陸野 礼子 （株式会社日新システムズ） （五十音順）

目次

1．はじめに

私たちのグループでは、

“「先人の知恵（智慧）」や「過去のトラブル対応経験」といったノウハウを有効に活用できるようになれば、組織全体の技術力向上に繋がり、過去と同じ失敗、あるいは類似の失敗を繰り返さない組織にすることができる。ひいては、「高品質なソフトウェアやシステムを開発・提供し続けられる組織」にして行ける。”
という仮説を立てました。

これまでも様々な方法でノウハウ共有が試みられましたが、多様な情報ソースを万人のニーズに合った形で提供することは困難でした。

その解決策として、昨今技術革新が進む**生成AIを活用すれば、利用者の期待に応える情報が提示できる**のではないかと考えました。

そのために、**組織のノウハウをどうすれば有効に活用できるのか、そして組織内の誰もが使いやすい仕組みにするにはどうすればよいか**について、討議を重ねてきました。

今回の発表では、私たちがこれらの討議の中で得た気付きをご紹介したいと思います。

・ナレッジ：知識や情報
・ノウハウ：ナレッジを基に経験を通して培ってきた知恵（智慧）
→ 単なるナレッジ活用ではなく、ノウハウ（智慧）を活用できるようにすることを目指す

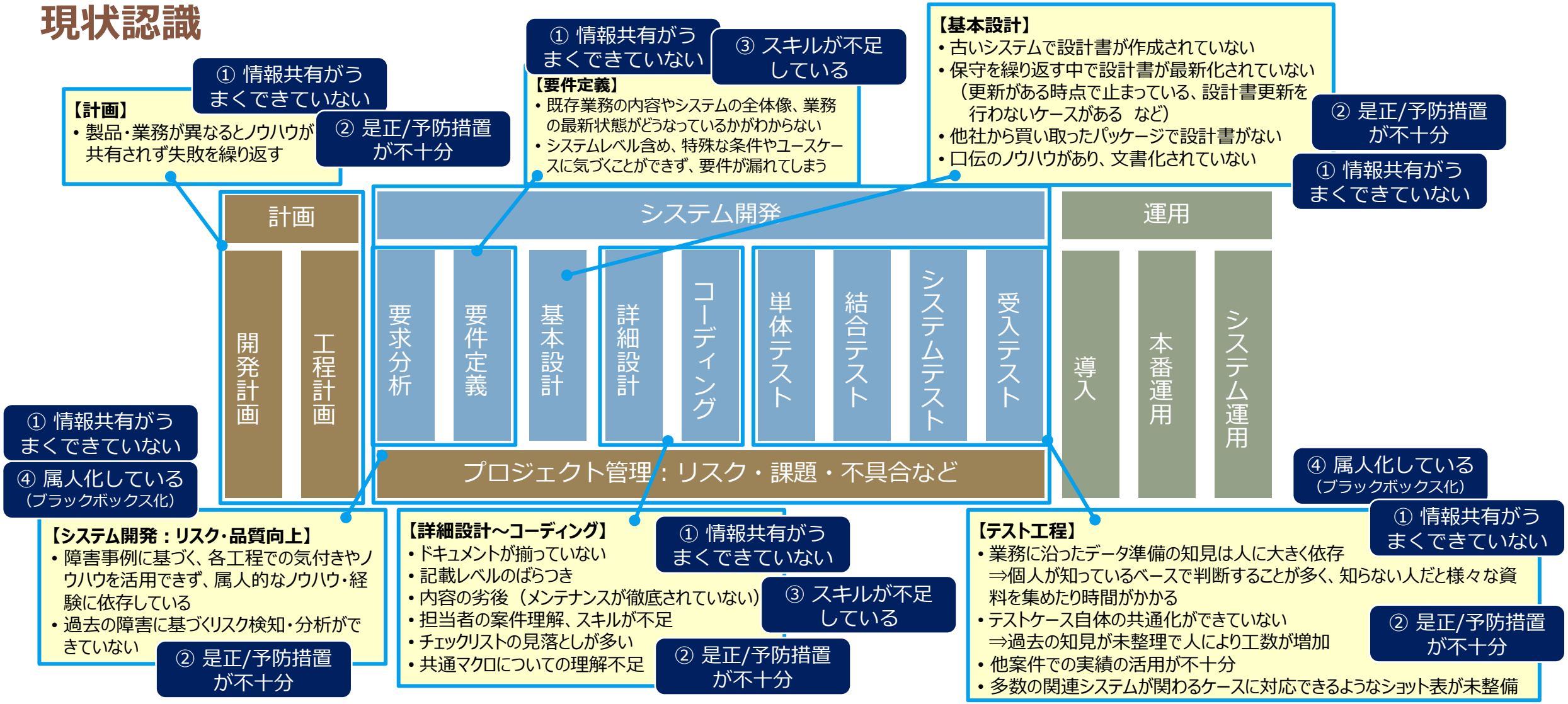
2．組織の課題と本活動のゴール

- **組織の課題**
  - これまで組織内のノウハウを共有するために様々な取り組みが試みられてきたが、いまだに一部の人だけが知っていることやできていることが共有されていなかったり、活用できなかったことなどに起因して、**過去と同じ失敗や類似の失敗が繰り返されている**
- **本活動のゴール**
  - 「先人の知恵（智慧）」や「過去のトラブル対応の経験」などのノウハウを有効に活用できるようにして、**失敗を繰り返さない組織にするための提案を導き出す**
  - **ゴールへのアプローチ**

組織の中には、失敗を繰り返さないノウハウを持っている人がいる → これまでに試みたノウハウ共有方法では多様な情報ソースを万人のニーズに合った形で提供することは困難 → **新たなノウハウ活用の戦略が必要**

開発工程と組織の課題

現状認識



新たなノウハウ活用の戦略の検討

戦略

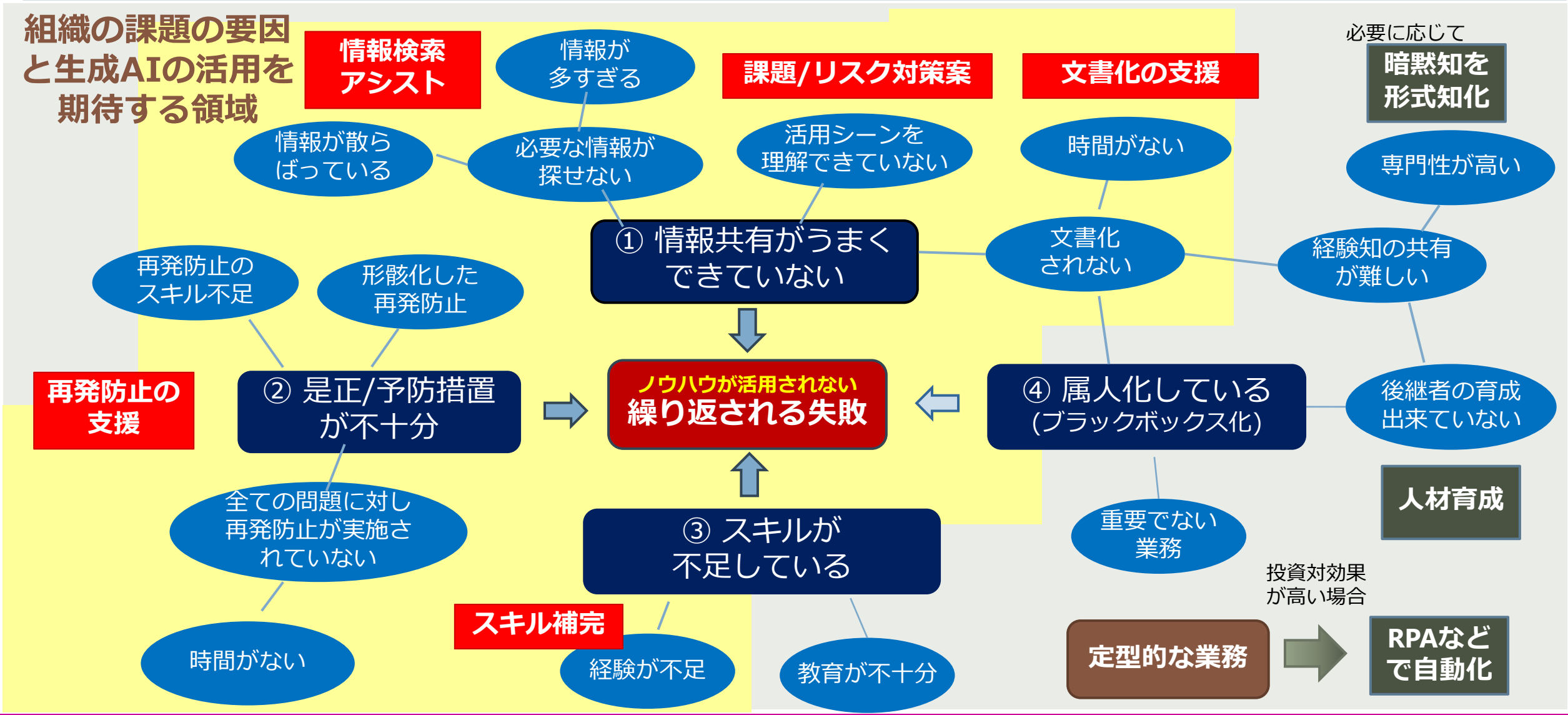
SWOT分析マトリクス

【 目標（あるべき姿）】

高品質なソフトウェアやシステムを開発・提供し続けられる組織

		内部環境	
		強み(Strength)	**弱み(Weakness)**
		・組織内にコア技術やドメイン特有の知識や知恵・経験をたくさん持っている技術者がいる ・高品質な製品を顧客に提供するために改善に取り組む組織風土・文化がある ・新しい技術を取り入れる柔軟性がある	・組織の一部の人しか知らない経験に基づく知識や技術が活用できていない ・人手不足、アウトソーシング頼り ・人材育成の仕組みが確立していない ・ノウハウを形式知化できていない ・過去と同様の失敗を繰り返している
外部環境	**機会(Opportunity)**	**強み×機会**	**弱み×機会**
	・生成AI(LLM)の急速な進化 ・DX化の拡大 ・働き方改革	・**生成AI**を使って組織のノウハウを有効に活用する技術を確立 ・組織のコア技術を効率的に残す・伝えるための**生成AI**を活用したDX推進 ・利用者個々人がオンデマンドで必要な情報を引き出せるシステムの提供	・**生成AI**による組織のノウハウから教育教材の自動生成と有識者による教育実施(プッシュ型人材育成) ・**生成AI**を活用したセルフレビューやチャットツールの提供(プル型人材育成) ・利用者個々人のシチュエーションに応じたナレッジが活用できるしくみ
	脅威(Threat)	**強み×脅威**	**弱み×脅威**
	・リモートワーク増加によるOJT機会の減少 ・2025年の崖(人材不足によりDXが進まない) ・情報セキュリティ対策	・**生成AI**を活用したセルフレビューやチャットツールの提供(プル型人材育成) ・オープンソースの**生成AI**を使ったメンテナンスが容易なノウハウ共有環境の提供 ・オンプレミス環境での構築やセキュリティが担保されているサービスを利用するセキュアなシステムの提供	・技術や技能を持った人材がいなくなる前に優先度が高い課題ごとに計画的に改善 ・経営層の意識を変えるために早く効果を出してDX人材の育成・獲得に積極的に投資させる ・**生成AI**の段階的導入(運用ルールの整備、セキュリティ対策が完了していて、簡単に使えるものから試行)

組織の課題と生成AIへの期待



改善のスコープ：品質向上への生成AIの活用

組織の課題

ノウハウが活用されない
繰り返される失敗

ノウハウ活用の対象は、「開発上流からの品質の作り込み」や「予防措置」なども含めた「プロセス品質」および「プロダクト品質」

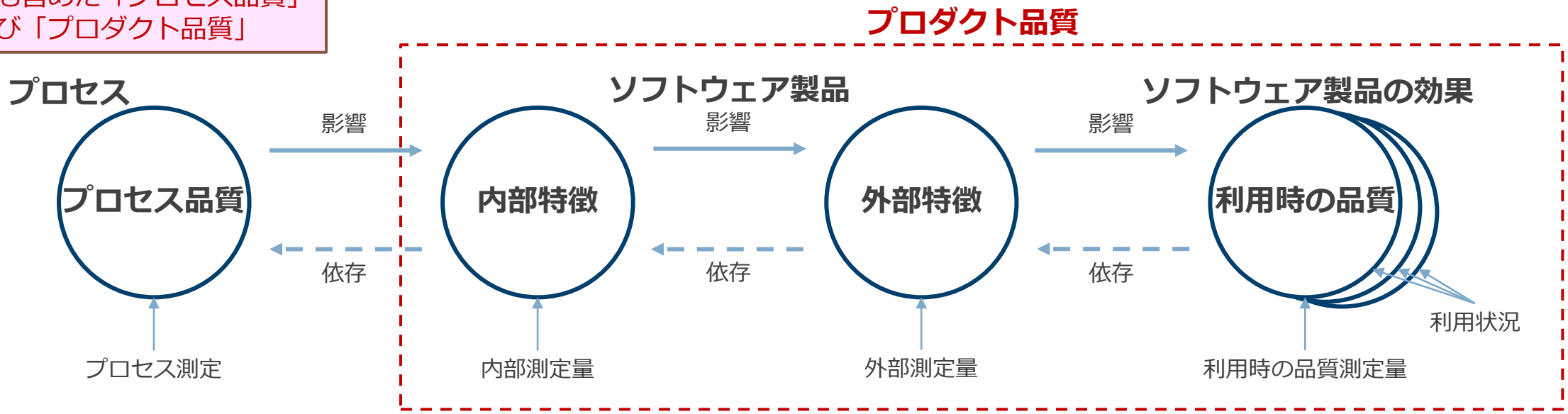
組織の課題の要因

① 情報共有がうまくできていない
② 是正/予防措置が不十分
③ スキルが不足している
④ 属人化している（ブラックボックス化）

生成AIへの期待

- 再発防止の支援
- 文書化の支援
- 情報検索アシスト
- スキル補完
- 課題/リスク対策案

“高品質なソフトウェアやシステムを提供し続けられる組織“ にするたに、生成AIの活用は生産性向上のためだけではなく、製品開発ライフサイクルにおけるプロセス、およびプロダクトの品質向上活動全般で考える



ソフトウェア製品のライフサイクルにおける品質

参考・出典：[ISO/IEC 25010 : 2011] JIS X 25010 : 2013, SQuBOK Guide V3

組織の課題に対する改善目標

各シーンにおける実践的生成AI活用（ユースケースからの検討）

【開発計画：プロジェクトリスクの提示】
- 過去案件の事例や教訓をノウハウとしてAIに学習させ、案件特性をインプットすることで、推進上発生しうるリスクを提示させる（品質保証の向上に寄与）

【要求分析】
- 過去の不具合事例をAIに学習させ、案件特性をインプットすることで、開発案件に関係する特殊な条件や考慮すべき観点などを提示させる

【コーディング：効率的な開発】
- コード解析ツールの指摘に対する修正案の提示
- AI技術を活用したコード生成
  ※どの程度まで利用可能かは分析評価を実施
- 自動テストによる効率化
- パラメータ不備などの単純ロジック以外のソースコードのチェックをAIで実現

組織の課題
- ① 情報共有がうまくできていない
- ② 是正/予防措置が不十分
- ③ スキルが不足している
- ④ 属人化している（ブラックボックス化）

計画：開発計画／工程計画

システム開発：要求分析／要件定義／基本設計／詳細設計／コーディング／単体テスト／結合テスト／システムテスト／受入テスト

運用：導入／本番運用／システム運用

プロジェクト管理：リスク・課題・不具合など

生成AIへの期待
- 再発防止の支援
- 文書化の支援
- 情報検索アシスト
- スキル補完
- 課題/リスク対策案

【工程計画：品質確保施策の提案】
- 過去に発生した不具合をAIに学習させ、PJ特性をインプットすることで、不具合に対する再発防止を提示させる
- チェックリストによるチェック

【設計工程：ドキュメントレビュー】
- チェックリストによる形式チェックをAIで実現（誤字脱字、仕様の曖昧性を含む）
- 不具合事例よりAIで根底となる観点を抽出
- その観点を入力としてAIで設計漏れを検知

【設計工程：設計書のUML生成】
- マークダウンで状態遷移図を描かせる

【プロジェクト管理：開発リスク・品質向上】
- 品質指標や生産性指標、リスクマネジメントのヒントをAIに提案させて品質戦略立案を支援
- 生成AIを並走させ、各工程で開発リスクや品質向上へのアドバイスをタイミングよく活用

【テスト工程：テスト設計】
- チェックリストによる形式チェックをAIで実現
- 状況に合ったテスト技法の提案やテストケース生成
- 過去障害をAIに学習させ、テストケースをインプットすることで、観点の過不足、追加ケースを提示させる

【テスト工程：品質分析支援】
- 発生した不具合の分類・傾向から品質向上策の観点を提示させる

４．仮説と研究課題(Research Questions)

仮説：

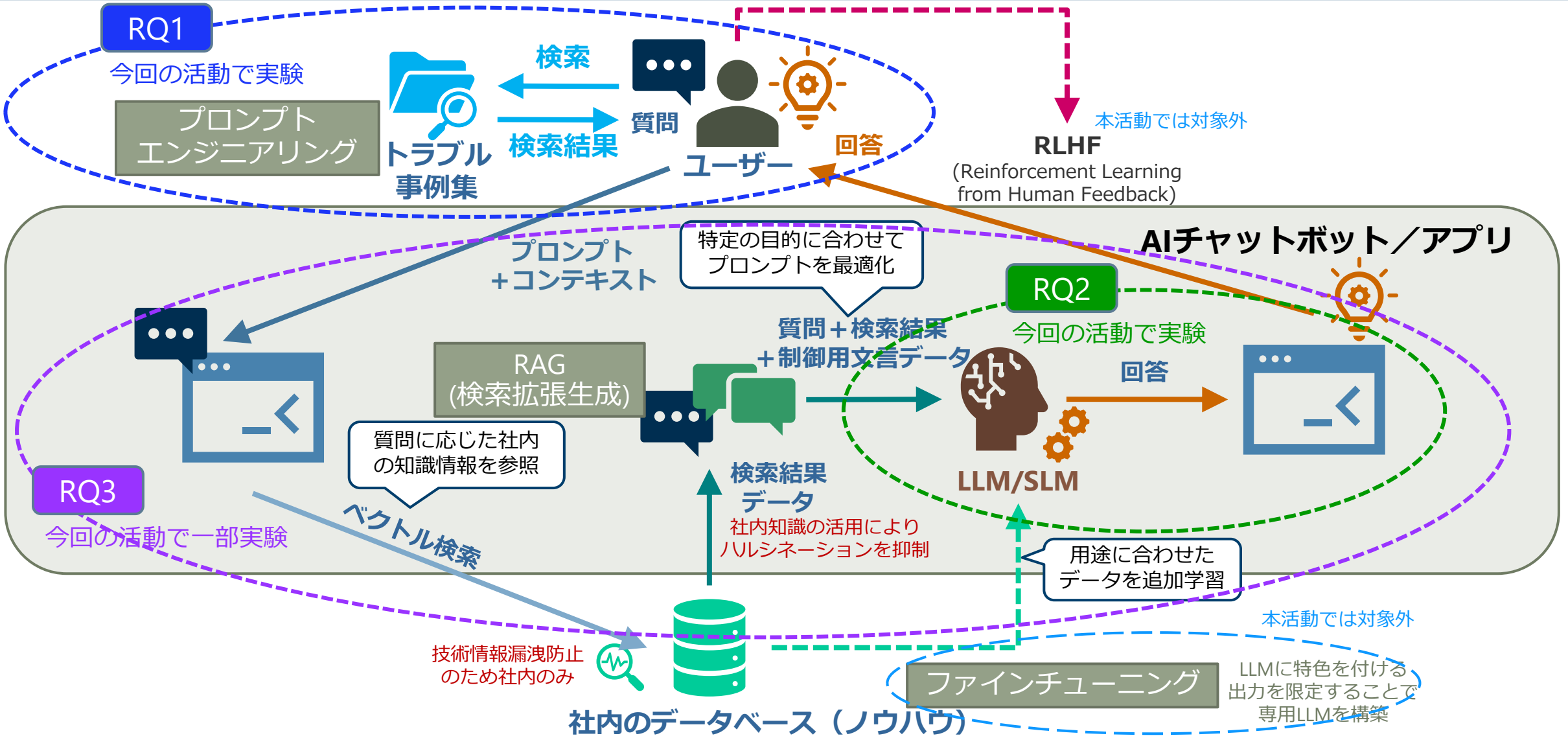
組織のさまざまなノウハウを活用したい場面において、ユースケースごとに生成AIの活用方法を工夫することで、万人のニーズに合った形でノウハウを提供・活用できる。

- **【RQ1】**
  ユースケースごとのプロンプト（生成AIへの入力・指示・質問）の内容を工夫すれば、特定の言語モデルを使わなくても、効果的に「先人の知恵（智慧）」や「過去のトラブル対応経験」といったノウハウを引き出して活用することができるか？

- **【RQ2】**
  用途に適したFoundation Modelを選定し、専用アプリから利用できるようにすれば、プロンプトエンジニアリングや追加学習などのテクニックを使わなくても、利用者のニーズに応えられるか？

- **【RQ3】**
  各組織のノウハウを活用できる方法を組合せることで、これまで試みてきた方法ではうまく行かなかったノウハウの共有・活用の課題が解決できるか？

業務に特化した生成AI活用の流れと研究課題（RQ）



組織の課題に対する改善目標（再掲）

各シーンにおける実践的生成AI活用（ユースケースからの検討）

★ 第15期活動の対象

【開発計画：プロジェクトリスクの提示】
★
- 過去案件の事例や教訓をノウハウとしてAIに学習させ、案件特性をインプットすることで、推進上発生しうるリスクを提示させる（品質保証の向上に寄与）

【要求分析】
★
- 過去の不具合事例をAIに学習させ、案件特性をインプットすることで、開発案件に関係する特殊な条件や考慮すべき観点などを提示させる

【コーディング：効率的な開発】
★ コード解析ツールの指摘に対する修正案の提示
★ AI技術を活用したコード生成
※どの程度まで利用可能かは分析評価を実施
- 自動テストによる効率化
- パラメータ不備などの単純ロジック以外のソースコードのチェックをAIで実現

組織の課題
- ① 情報共有がうまくできていない
- ② 是正/予防措置が不十分
- ③ スキルが不足している
- ④ 属人化している（ブラックボックス化）

計画
- 開発計画
- 工程計画

システム開発
- 要求分析
- 要件定義
- 基本設計
- 詳細設計
- コーディング
- 単体テスト
- 結合テスト
- システムテスト
- 受入テスト

運用
- 導入
- 本番運用
- システム運用

プロジェクト管理：リスク・課題・不具合など

生成AIへの期待
- 再発防止の支援
- 文書化の支援
- 情報検索アシスト
- スキル補完
- 課題/リスク対策案

【工程計画：品質確保施策の提案】
- 過去に発生した不具合をAIに学習させ、PJ特性をインプットすることで、不具合に対する再発防止を提示させる
- チェックリストによるチェック

【設計工程：ドキュメントレビュー】
★
- チェックリストによる形式チェックをAIで実現（誤字脱字、仕様の曖昧性を含む）
★
- 不具合事例よりAIで根底となる観点を抽出
★
- その観点を入力としてAIで設計漏れを検知

【設計工程：設計書のUML生成】
★
- マークダウンで状態遷移図を描かせる

【プロジェクト管理：開発リスク・品質向上】
★
- 品質指標や生産性指標、リスクマネジメントのヒントをAIに提案させて品質戦略立案を支援
- 生成AIを並走させ、各工程で開発リスクや品質向上へのアドバイスをタイミングよく活用

【テスト工程：テスト設計】
★
- チェックリストによる形式チェックをAIで実現
★
- 状況に合ったテスト技法の提案やテストケース生成
- 過去障害をAIに学習させ、テストケースをインプットすることで、観点の過不足、追加ケースを提示させる

【テスト工程：品質分析支援】
★
- 発生した不具合の分類・傾向から品質向上策の観点を提示させる

5．実験

- 【RQ1】
  ユースケースごとのプロンプト（生成AIへの入力・指示・質問）の内容を工夫すれば、特定の言語モデルを使わなくても、効果的に「先人の知恵（智慧）」や「過去のトラブル対応経験」といったノウハウを引き出して活用することができるか？

- 【RQ2】
  用途に適したFoundation Modelを選定し、専用アプリから利用できるようにすれば、プロンプトエンジニアリングや追加学習などのテクニックを使わなくても、利用者のニーズに応えられるか？

- 【RQ3】
  各組織のノウハウを活用できる方法を組合せることで、これまで試みてきた方法ではうまく行かなかったノウハウの共有・活用の課題が解決できるか？

※ 情報セキュリティの考慮が必要な実験データを利用するケースでは
- オンプレミス環境に構築し、Closed Network環境またはスタンドアロンで評価
- モデルをダウンロードして利用するが、インターネットには何も情報をアップロードしない

一般的な情報のみを使った一部の実験では、Saasのサービス（ChatGPT、Google Gemini、Microsoft Copilotなど）も利用する

実験１

【RQ1】ユースケースごとのプロンプト（生成AIへの入力・指示・質問）の内容を工夫すれば、特定の言語モデルを使わなくても、効果的に「先人の知恵（智慧）」や「過去のトラブル対応経験」といったノウハウを引き出して活用することができるか？

- 実験
  - 各組織で使える生成AIの環境でプロンプトを少し工夫してみると、期待に応える回答に近づけるかを確認する（有効なキーワードなどを見つけられないか？）
- 結果
  - **「組織の課題に対する改善目標： 各シーンにおける実践的生成AI活用**（P.12）**」に挙げたユースケースについて、プロンプトの書き方を工夫することで、単なる検索に比べて期待する質の高い回答を導き出せることが試せた**
    （具体的なユースケースごとの活用例は、16期で報告予定）
    - 立場や、回答内容などを具体的に指示する、
      指示をステップに分割する、参照テキストなどを提供
    - わかりやすく、短くシンプルに、より具体的に
    - 回答が得られない場合、言葉を変えてみる
- **考察**
  - **テンプレートを用意する**
    参考になる文例が掲載されているWebサイトがたくさんある
  - **専用のUIを用意する**
    簡単なアプリ開発のフレームワークなどを利用する
  - **精度向上のために次ページのような技術を試す**

プロンプト記載例：

初心者にもわかりやすく説明してください。

注意すべきポイントを指摘してください。

メリットと、デメリットを列挙してください

間違いがある場合、修正例を示してください

「○○○」という言葉を使って、・・・

「△△△」という表現は絶対に使わずに、・・・

理由、根拠、具体例を含んで、・・・

○○の立場で、・・・

足りない要素があれば補いながら、・・・

実験１　代表的なプロンプトエンジニアリング技術について

（実験１では調査のみ・実験３で試行）

1. **Few-Shotプロンプティング**
   - プロンプト内のデモを提供してモデルをより高い性能に導く文脈学習を可能にするテクニック
2. **Chain-of-Thought（CoT：思考連鎖）プロンプティング**
   - 中間的な推論ステップ（推論の連鎖）を介して複雑な推論能力を可能にする
     few-shot promptingと組み合わせることで、推論が必要なより複雑なタスクでより良い結果を得ることができる
3. **Self-Consistency（自己整合性）**
   - few-shot CoTを通じて複数の多様な推論パスをサンプリングし、生成物を使用して最も整合的な回答を選択する
4. **Retrieval Augmented Generation（RAG：検索拡張生成）**
   - 生成AIが質問に回答する際に、生成AIのデータベースに加え、膨大な自社のデータベースから情報を検索して回答させるように自社データを組み込むテクニック
5. **ReAct（推論）＋（行動）**
   - Reasoning（推論）と Acting（行動）を組み合わせた造語で、外部ツールと対話して追加情報を取得し、より信頼性の高い事実に基づく回答を生成するテクニック

参考：[Prompt Engineering Guide | Prompt Engineering Guide (promptingguide.ai)]

基盤モデルが持っていない組織の知識を上記の方法で追加するために、組織としてどのような形でデータを蓄積して行くかを検討する
（上記の１と４について実験３の中で試行した）

実験２

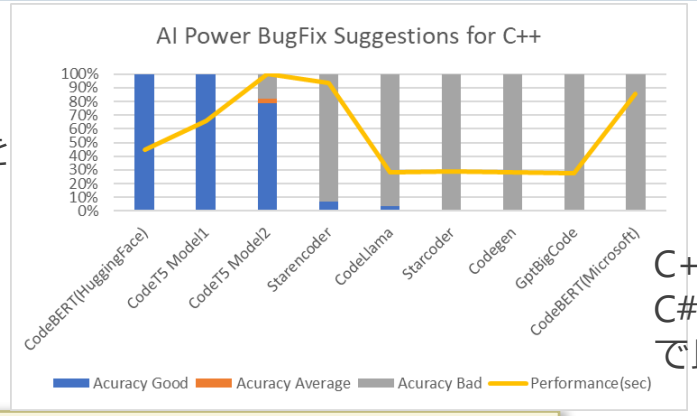
【RQ2】用途に適したFoundation Modelを選定し、専用アプリから利用できるようにすれば、プロンプトエンジニアリングや追加学習などのテクニックを使わなくても、利用者のニーズに応えられるか？

- 実験
  - ① コード解析ツールの指摘内容の修正に適したオープンソースのLLMの評価
  - ② 専門分野のノウハウ回答用チャットボットに適したLLMの評価
  - ③ オンプレミス環境で日本語ドキュメントのレビュー用に使えるSLM（Small Language Model）の評価
- 結果
  - ① コード修正提案について9つのLLMを評価(次ページで説明) → Q&Aを繰り返さずに修正案を提示できた
    - **C++用：CodeBERT (HuggingFace)、C#用：CodeT5 Model1 で良好な結果が得られた**（C++での実験結果を次ページに掲載）
  - ② AIチャットボット用に7つのLLMを評価
    - **英語：Llama3, 日本語：Mistral-7B-Instruct-v0.2 で良好な結果が得られた**
      - チャットボット用に7つのオープンソースのLLMを評価（Meteor Score, Rouge Score, 人による正確性の判定）を実施
        Mistral-7B-Instruct-v0.2, Mistral-7B-Instruct-v0.3, Llama-3-8B-Instruct, Llama-2-7b-chat-hf, stablelm2-1_6b-chat, falcon-7b, flan-t5-large-grammar-synthesis
  - ③ 日本語のドキュメントレビューに使える4つの自然言語処理に強いSLM（Small Language Model）を評価
    - **実運用では高性能なCPU、GPU、16GB以上のRAMを搭載したワークステーションが必要。日本語の処理能力は良好。**
      - Llama -2-Swallow-7b-instruct-v0.1, Llama-3-Swallow-8b-instruct-v0.1, Phi-3-mini-4k-instruct, GiNZA v5.2.0 を評価（簡単な評価のみ。詳細調査は継続中。）
        Llama -2ベースのSwallowでは良好な結果。 Llama-3ベースのSwallowは7/1にリリースされたばかりなのでまだ評価中。Phi-3は良好な結果が得られず。
- **考察**
  - **万人向けには、利用目的に合ったモデルを選定、専用アプリを通して利用することで、生成AI初心者が利用しても、期待する答えを早く引き出せる可能性がある**

実験２ コード修正提案アプリでの実験

９つのOSSのLLMをベンチマーク

1008の過去の指摘修正データをPickle化して28の指摘の修正について正確性、処理時間を評価



C++用：CodeBERT (HuggingFace)
C#用： CodeT5 Model1
で良好な結果が得られた

① Code

C++
C#

② Analyze

coverity

解析レポートとソースコードをアップロード

③ Visualize

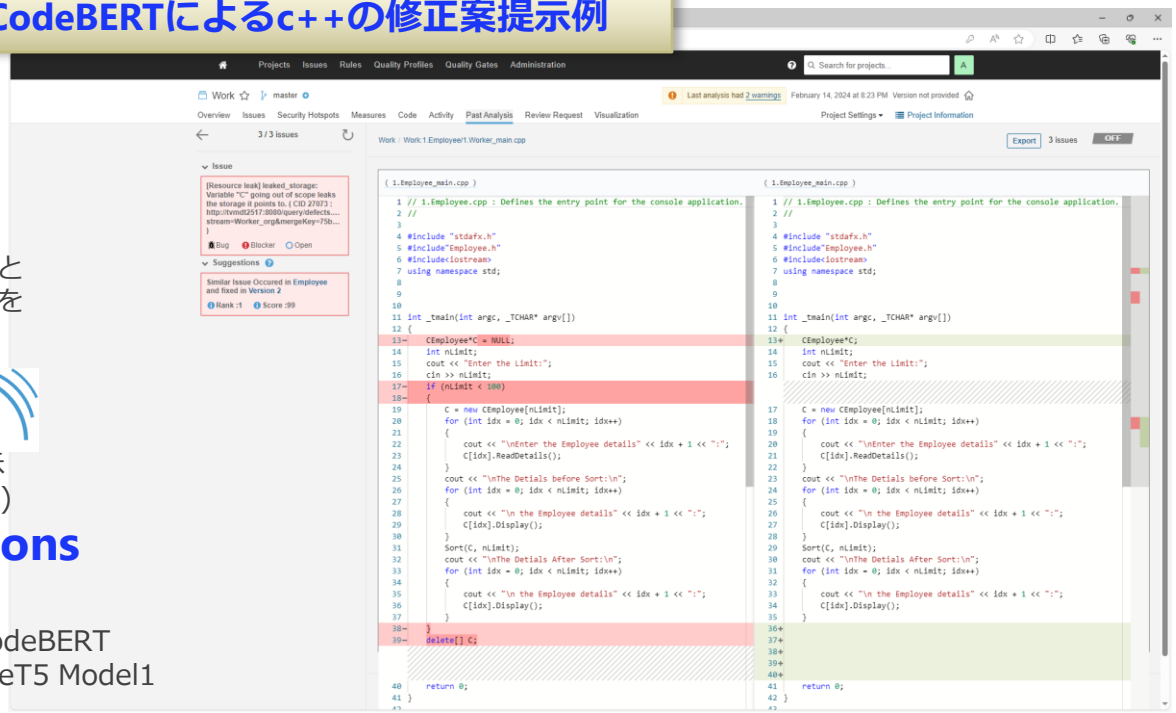
sonarqube

解析結果表示
(独自Plug-in)

④ Bug fix suggestions

C++: CodeBERT
C#: CodeT5 Model1

CodeBERTによるc++の修正案提示例

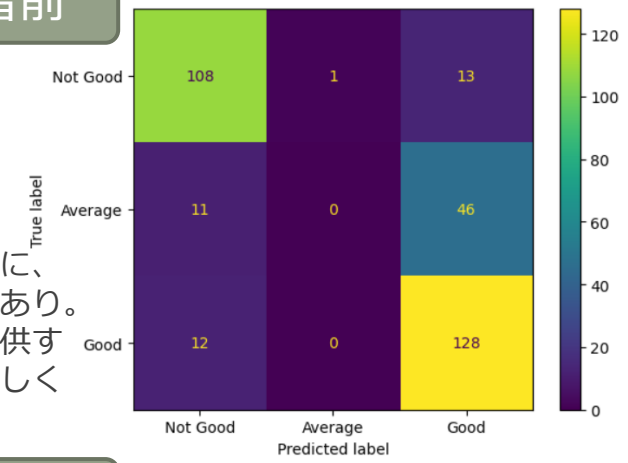


Embeddingによる追加学習（前処理）

混同行列（Confusion Matrix）による評価：CodeBERT

（347のコードブロックで実験）

追加学習前



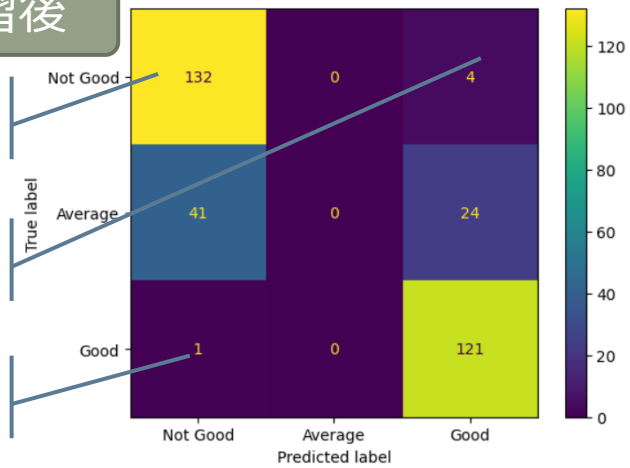
精度向上のために、追加学習は効果あり。専用ツールを提供すれば、運用は難しくない。

追加学習後

真陽性(TP)が増えた

擬陰性(FN)が減った

擬陽性(FP)が減った



実験３

【RQ3】各組織のノウハウを活用できる方法を組合せることで、これまで試みてきた方法ではうまく行かなかったノウハウの共有・活用の課題が解決できるか？

- 実験
  - ① 組織のノウハウを活用できる環境：実験１で記載したFew-Shotプロンプティングを試行
  - ② 組織のノウハウを活用できる環境：実験１で記載したRAG (Retrieval-Augmented Generation) を試行
  - ③ 組織データの蓄積の方法の検討

  ※ 特定のケースでのコードのBugfixの知識追加では、Embedding（CSV fileをベクトル化してLLMに追加学習）で効果を確認できた（実験２）

- 結果
  - ① 組織のRedmine DBの情報をFew-shot promptingを使ってチャットボットにインプットして評価したが良好な結果は得られなかった（**データの質に問題があると考えられる**）
  - ② **RAGの評価：Faiss（類似ベクトル検索ライブラリ）を実験２の③のSLM(Swallow)で試行した**
    - **MS-Wordの要求仕様書(90ページ程度)を使用し、簡単な文章のチェックができることを確認。精度の評価は継続中。**
  - ③ RAGの性能向上のためには、ドメイン固有のデータを収集する仕組み、収集データからノイズを除去して利用するモデルに適した形式にする仕組み、利用者のフィードバックを集め改善などを行う必要がある

- **考察**
  - **回答精度を向上させる（質問回数・誤回答を減らす）には、組織のローカルデータ（特定の知識）の品質を向上させる必要があるが、従来と同じような運用ではメンテナンスされずに活用できないデータになってしまう可能性が高いため、簡単なタグ付けや大括りでの分類程度だけでもやれる仕組みを作ると良い**
    → 引き続き、データ格納時の工夫、検索時の工夫、回答生成時の工夫について調査や試行を続けて行く

6．活動の結果と考察・気づき

- 活動の結果
  - これまでの活動では、各組織の課題について限定的なユースケースのみでの試行となった。
  - 目的に特化した生成AI（モデル）を、専用のアプリケーションやプロンプトのテンプレートを備えたチャットボットを通して利用することで、生成AIの初心者であっても、必要な情報やノウハウを適切なタイミングで取得できるため、開発や品質保証活動において大きな助けとなる可能性を示すことができた。

- 考察
  - 目的に合ったモデルを選び、プロンプトの書き方を工夫するだけでも、比較的良い結果を得ることができた。
  - EmbeddingsやRAGを組み合わせることで、学習済みモデルの知識に組織のノウハウを追加し、利用者の多様なニーズに応える能力を持たせることが可能である。このアプローチにより、モデルは（学習済みの）既存の知識を活用しつつ、特定の業界や分野に特化した情報を提供することができる。
    しかし、回答精度の向上には、組織のデータの質の向上が重要な課題となる。
    高品質なデータを用いることで、モデルはより正確で信頼性の高い情報を生成することができ、最終的には利用者の満足度を高めることに繋がる。

6．活動の結果と考察・気づき（つづき）

- 気づき：
  - ユーザの利用場面とユースケースから考えるアプローチは良かった。
  - ヒットしたWebサイトのリンクを出すだけの検索エンジンとは違い、知りたい情報を読みやすく要約して表示してくれる（プロンプトでちゃんと伝える必要あり）ところは、とても助かる。
  - 生成AIの回答精度は100%ではないので、最終的に人が結果の妥当性を確認する必要がある。そのためにも生成AIに何でも頼るのではなく、目的の明確化や回答を正しく活用できるような技術スキルも備えておくことも必要。
  - こういう回答を引き出したいという想いをうまくプロンプトに入力できていないので、より多くの経験を積んでコツをつかんで行く（継続して活動する）。
  - ナレッジ（知識や情報）だけではなく、ノウハウ（ナレッジを基に経験を通して培ってきた智慧）を活用するためには、プル型（オンデマンド）で利用者の都合の良いタイミングや困っている場面でいつでも利用できるという良い面もあるが、プッシュ型（ハンズオン教育やワークショップなどの体験型学習やOJTなど）も併用すべきである。
    - 教育用のテキスト作成などにも、生成AIが活用できる
  - 実運用では、生成AI利用のリスク対処方法の検討が必要。
    - 機密情報の漏洩、ハルシネーション（誤答）、AIの中身がブラックボックスであること、生成物の著作権の問題 など

７．まとめ・今後の活動

- まとめ
  - 生成AIのモデルを追加学習で性能向上させるよりも、最良のFM（基盤モデル）を選定したり、最新版にアップグレードした方が良い結果が得られた。
    メジャーなLLM/SLMは頻繁に新しいバージョンがリリースされるので、利用しているモデルが最新版かどうか・最新版でどんな点が改良されたかをチェックしておくことを推奨する。
  - それぞれのモデルには、得意／不得意がある。
    たとえば、チャットボット向きのモデルや、特定の開発言語のコード生成に強いモデルがある。
    また、モデル回答の精度が良くてもパフォーマンスでは他のモデルに劣る（回答速度が遅い）ものや、事前準備に多くの時間を要するモデルもある。
    日本語文書の作成やレビューには、日本語処理に強い（事前学習済み）モデルを利用する。
- 今後の活動（第１６期の活動内容：予定）
  - 改善すべき点：15期では生成AIの利用環境の提供が不十分だったり、利用上のルール整備が整っておらず、思ったような調査や検証ができなかった。早い段階で実験環境の整備を進める。
  - 回答精度の向上のためのいろいろなテクニックについても、より深いレベルの評価を行う。
  - 対象のスコープで挙げた多くのユースケースについて、生成AIを活用したノウハウの提供による各組織の課題解決に対する効果を検証し、運用プロセスを策定する。

ご清聴ありがとうございました