HITACHI

Hitachi Group Company

生成AIを活用したレビュー効率の向上と有効性の検証

株式会社 日立製作所
デジタルシステム&サービス統括本部
品質保証統括本部 社会システム品質保証本部
公共システム品質保証部
第1G
西本真由

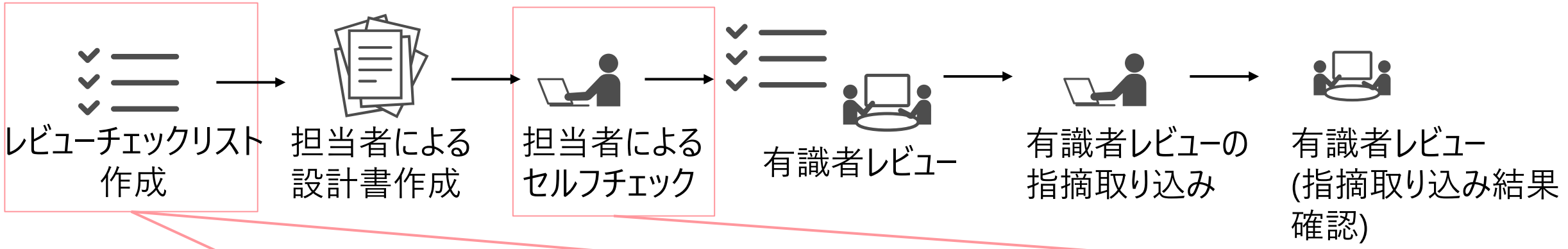
Contents

Contents

1.1 概要

ソフトウェア品質を向上させる取り組みとして
レビューチェックリストを用いたレビューの実施を検討する際
以下のような課題に悩んでいませんか？

■(例)設計工程の開発プロセス



課題1

レビューチェックリストへの
フィードバックは時間がかかる
⇒メンテナンスが追い付かない

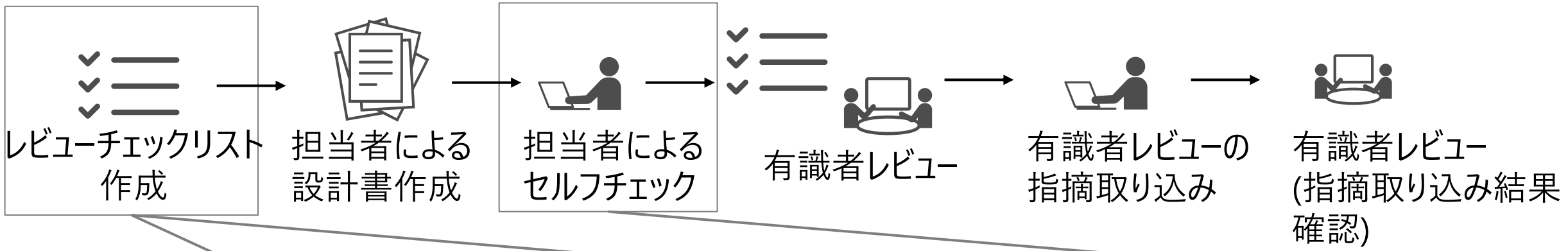
課題2

暗黙知やプロジェクト独自の観点を
含める必要がある
⇒レビューチェックリストの質
(スキル依存)

課題3

レビュー成果物と
レビューチェックリストの数が膨大
⇒記載箇所特定/妥当性評価に
工数が膨大

1.1 概要



課題1

レビューチェックリストへのフィードバックは時間がかかる
⇒メンテナンスが追い付かない

課題2

暗黙知やプロジェクト独自の観点を含める必要がある
⇒レビューチェックリストの質(スキル依存)

課題3

レビュー成果物とレビューチェックリストの数が膨大
⇒記載箇所特定/妥当性評価に工数が膨大

解決策

生成AIにレビューチェックリストを作成してもらう

解決策

生成AIにセルフチェックをしてもらう

課題1,2説明（2章：スライド8~18）

課題3説明（3章：スライド19~25）

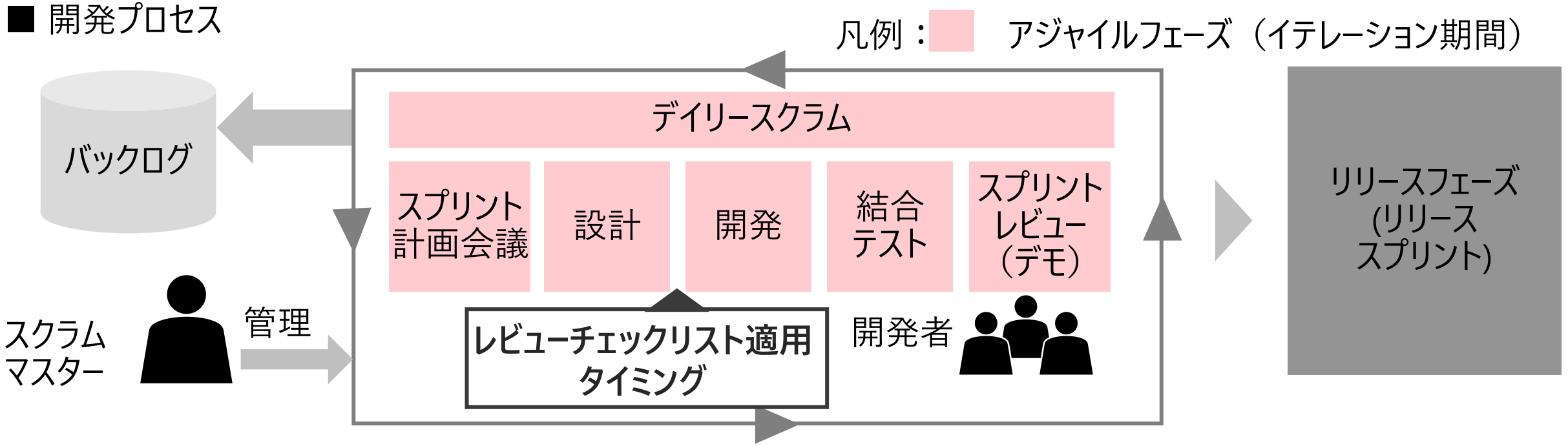
1.2 前提（適用案件の概要、使用データ）(1)

HITACHI

■アジャイル開発のAプロジェクト

- Aプロジェクトでは半年ごとに既存機能の改修・新規機能実装を実施
- スプリント期間は 2 週間
- 今回開発の開発規模は14.5kS
- 2024/n月～(約6か月)の開発でレビューチェックリストを適用

■ 開発プロセス

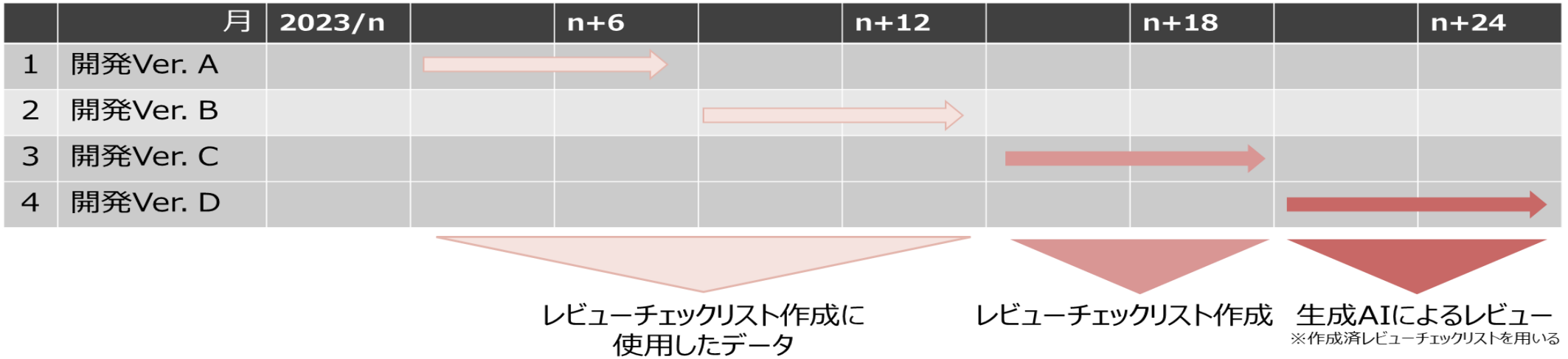


1.2 前提（適用案件の概要、使用データ）(2)

HITACHI

Point

過去開発のバグをインプットにレビューチェックリストを作成し、
作成したレビューチェックリストをインプットに生成AIによるセルフチェックを実施



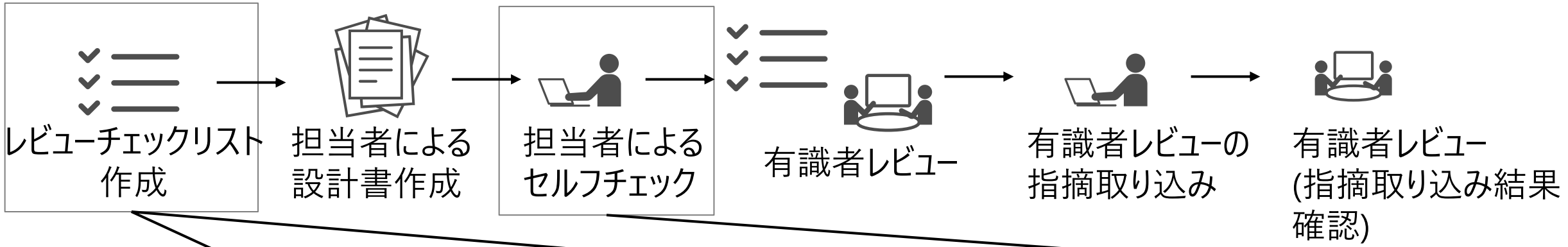
■レビューチェックリスト作成での生成AIへのインプットデータ
- 過去の開発2回分（開発期間は2023/n月～(1年分)）
- 設計レビューのバグを利用（バグ件数は26件）

■生成AIによるセルフチェックでの生成AIへのインプットデータ
- 生成AIを用いて作成したレビューチェックリスト、社内の汎用ナレッジ
- 最新の設計書

■使用ツール
- Webブラウザ上で動作する生成AIサービスを利用（当社とグループ会社専用サービス）

Contents

2.1 課題と対応策（再掲）



課題1	課題2	課題3
レビューチェックリストへのフィードバックは時間がかかる ⇒メンテナンスが追い付かない	暗黙知やプロジェクト独自の観点を含める必要がある ⇒レビューチェックリストの質(スキル依存)	レビュー成果物とレビューチェックリストの数が膨大 ⇒記載箇所特定/妥当性評価に工数が膨大
解決策		**解決策**
生成AIにレビューチェックリストを作成してもらう		生成AIにセルフチェックをしてもらう

課題1,2説明（2章：スライド8~18）

課題3説明（3章：スライド19~25）

2.1 課題と対応策

HITACHI

課題1	課題2
レビューチェックリストへのフィードバックは時間がかかる ⇒メンテナンスが追い付かない	暗黙知やプロジェクト独自の観点を含める必要がある ⇒レビューチェックリストの質(スキル依存)

自然言語処理を用いて過去のバグを要約および抽象化し、その結果をもとにレビューチェックリストを作成する取り組みが研究されている[1]

目的

先行研究の手順を生成AIを活用することで自動化すれば効率的にレビューチェックリストを作成できるだろうか…？

⇒実際にプロジェクトに適用して効果を確かめる

[1] 渡邊正隆, 森崎修司, 松本健一 (2010). バグ報告の単語出現頻度に着目したチェックリスト作成の試行. 研究報告ソフトウェア工学 (SE), 2010(30), 1-7.

2.2 施策の詳細と検証内容

HITACHI

先行研究の手順と今回の取り組みでの変更点・工夫点

#	先行研究の手順	本取り組みでの手順
1	すべてのバグ報告から単語を抽出する。	プロンプトa)バグ一覧から単語を抽出する。出現頻度が高い順に並び替え、単語リストを作成する。
2	単語の出現頻度を計算し、出現頻度が高い順に並び替える。出現頻度別の単語リストを作成する。	
3	作成した単語リストの上位にある単語を含むバグ報告をすべてのバグ報告から抽出する。	プロンプトb)単語リストにある単語を含むバグをバグ一覧から抽出する。
4	バグを要約する。	
5	バグを抽象化する。	プロンプトc)バグを抽象化する。
6	抽象化したバグをグルーピングする。	削除。

2.2 施策の詳細と検証内容

HITACHI

#	先行研究の手順	本論文での手順
1	すべてのバグ報告から単語を抽出する。	プロンプ… …する。
2	単語の出現頻度を計算し、出現頻度が高い順に並び替える。出現頻度別の単語リストを作成する。	
3	作成した単語リストの上位にある単語を含むバグ報告をすべてのバグ報告から抽出する。	プロンプトb)単語リストにある単語を含むバグをバグ一覧から抽出する。
4	バグを要約する。	
5	バグを抽象化する。	プロンプトc)バグを抽象化する。
6	抽象化したバグをグルーピングする。	削除。

【変更点1】上位の単語ではなく、すべての単語を抜き出し

【変更点2】グルーピングはしない

変更理由

知見

- 少数しかない**重要なバグを見逃す恐れ**
- アジャイル開発で規模が大きくない案件で生成AIへのインプットとなる指摘件数も少ない
  ⇒最終的なアウトプットを手作業で確認しても大きな工数はかからない

バグ件数が多い場合：バグの重要度区分を参照させ重要度が高いものはすべての単語を抜き出す

2.2 施策の詳細と検証内容

HITACHI

	課題	検証内容
課題1	レビューチェックリストへのフィードバックは時間がかかる ⇒メンテナンスが追い付かない	手作業で実施する際と生成AIを用いる場合での作業時間の比較 【目標】作業時間：50％減
課題2	暗黙知やプロジェクト独自の観点を含める必要がある ⇒レビューチェックリストの質(スキル依存)	①手動で作成したレビューチェックリストを基準に、生成AIに作成させた場合の再現度 【目標①】再現度：70％以上 ②下記のパターンにおける指摘件数とバグ件数※1の比較 AIで作成したCLをレビューに適用した場合 VS で作成したCLをレビューに適用した場合 【目標②】下記表に記載

前回同様の品質と仮定し、
対象ページ数の減少に合わせ、
前回開発の**3割を目標**に

#	対象頁数	指摘件数	バグ件数
AI 今回開発	27	5	1
前回開発※2	90	16	3

※1 指摘件数:バグの疑いがありバグ一覧に起票された件数、バグ件数:指摘件数のうちバグであると確定した件数 ※2 開発Ver.B

2.3 有効性検証結果と評価（作業時間の削減効果）

HITACHI

Point

生成AIを利用時は手作業に比べ作業時間が1時間以上削減でき、比率にすると約50%作業時間を削減

#	比較条件	レビューチェックリスト作成	評価
1	手作業(先行研究の手順を手作業)	2h35min	−
2	生成AI利用	1h00min	○

[凡例(評価欄)]　◎目標の倍以上、○目標どおり(作業時間半減)、×目標を下回る(仮説で定義した数値の半分以下)、−対象外

考察

✓ 良かった点

内容を整理する、入力するなど、**手作業がほぼ無い**ため作業時間が削減できた

大規模な開発プロジェクトに対して適用する場合は生成AIを用いることで、**さらに作業時間の削減効果が期待できる可能性**がある

- 大規模なプロジェクトではバグ件数が多い傾向
  本論文の手法を手作業で実施するとさらなる工数がかかることが想定される
  (今回:14.5kSの開発規模、インプットバグ件数は26件)

2.3 有効性検証結果と評価（レビューチェックリストの質）

HITACHI

■ 生成AIを利用してレビューチェックリストを作成した場合の精度

項目	再現率	評価
再現率（手動で作成したチェックリスト全観点に対してAIが再現した割合）	89%	○

[凡例(評価欄)]　◎目標を上回る(90%以上)、○目標(70%)どおり、×目標を下回る(50％以下)、−対象外

考察

良かった点

レビューチェックリストを作成する場合暗黙知やプロジェクト独自の観点が含まれ、知識が必要不可欠

▶ 本手法を利用することで、知識レベルの不足などの理由による**レビューチェックリスト作成のハードルを下げる**ことができる

問題点

完全再現には至らなかった。
再現できなかったパターンは全て過去のレビューでバグとして未摘出(作りこみ無し)
バグ一覧をインプットとしているため、**過去に摘出したバグの内容に依存**する

▶ 作成するレビューチェックリストの品質を向上するには、バグ一覧のみならず、設計情報や、社内の標準レビュー観点表などを参照させる

2.3 有効性検証結果と評価（レビューチェックリストの質）

HITACHI

Point

指摘やバグの件数は、前回開発と同等の水準

■ 生成AIによって作成したレビューチェックリストを用いて実際にレビューした結果

項目	対象頁数	指摘件数※1	バグ件数※1	バグ密度※2	評価
過去開発の実績	90	16	3	3.3	–
仮説	27	5	1	3.7	–
実績	27	4	1	3.7	○

※1指摘件数は、バグの疑いがありバグ一覧に起票された件数を示し、バグ件数は指摘件数のうちバグであると確定した件数を示す
※2 100頁あたりの不良件数
[凡例(評価欄)]　◎目標の倍以上、○目標どおり、×目標を下回る(仮説で定義した数値の半分以下)、−対象外

考察

問題点

定量的にはおおよそ目標どおりであるが仮説よりも実際の指摘件数が少ない。
本手法を用いて作成したレビューチェックリストを適用した今回開発は、過去2回分の開発と開発内容の傾向が一部違った。

- インプット予定のバグ一覧と適用したいプロジェクトの開発内容にかい離がある場合は、類似の機能を実装した別のプロジェクトのバグ一覧を流用する

2.3有効性検証結果と評価（Aプロジェクトと異なるタイミングでの適用）

HITACHI

Aプロジェクトの設計工程に適用した結果、手法の有効性が確認できた!
他のプロジェクトや他の工程(テストや操作マニュアル他)に適用できそう

[アジャイル開発] Bプロジェクトの課題

- イテレーション期間で実システムを操作した結果バグが出た
- 同様のバグがないかを見直しすべき
- 見直し観点をバグ一覧から作成したい

リリースPhの品質向上観点として使用し、ソフトウェアの品質担保に貢献できた

2.3有効性検証結果と評価（Aプロジェクトと異なるタイミングでの適用）

HITACHI

■ AプロジェクトとBプロジェクトの条件差異

項目	Aプロジェクト	Bプロジェクト
使用データ	過去開発2回分のバグ	アジャイルPhでのバグ
適用タイミング	今回開発(設計)	リリースPh
インプットデータ量(バグ件数)	26件	17件

良かった点

適用タイミングが違う、プロジェクトが違っても適用でき、作業工数とレビューチェックリストの質に関する課題が解決できた

- アジャイル開発においても効率的な品質分析(弱点分析)からのフィードバックによる品質確保が可能

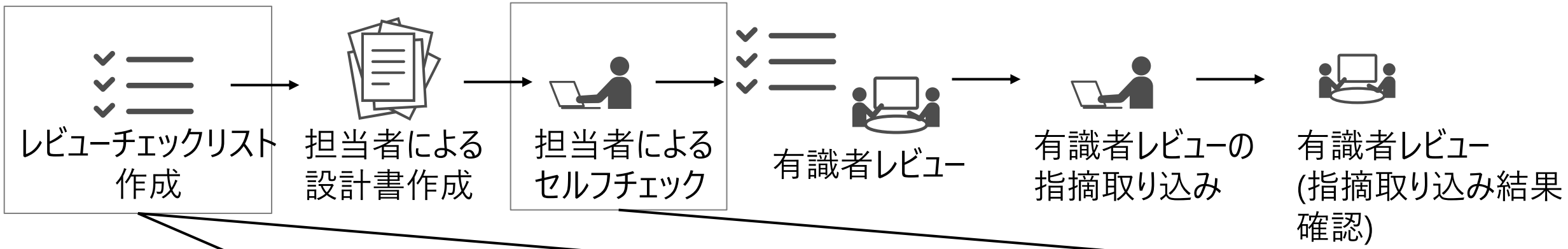
対策

バグとして摘出されていない観点はレビューチェックリストに出てこない

- セキュリティに関する観点や重要機能に対する観点を手動で追加

Contents

3.1 課題と対応策（再掲）



課題1

レビューチェックリストへのフィードバックは時間がかかる
⇒メンテナンスが追い付かない

課題2

暗黙知やプロジェクト独自の観点を含める必要がある
⇒レビューチェックリストの質(スキル依存)

課題3

レビュー成果物とレビューチェックリストの数が膨大
⇒記載箇所特定/妥当性評価に工数が膨大

解決策

生成AIにレビューチェックリストを作成してもらう

解決策

生成AIにセルフチェックをしてもらう

課題1,2説明（2章：スライド8~18）

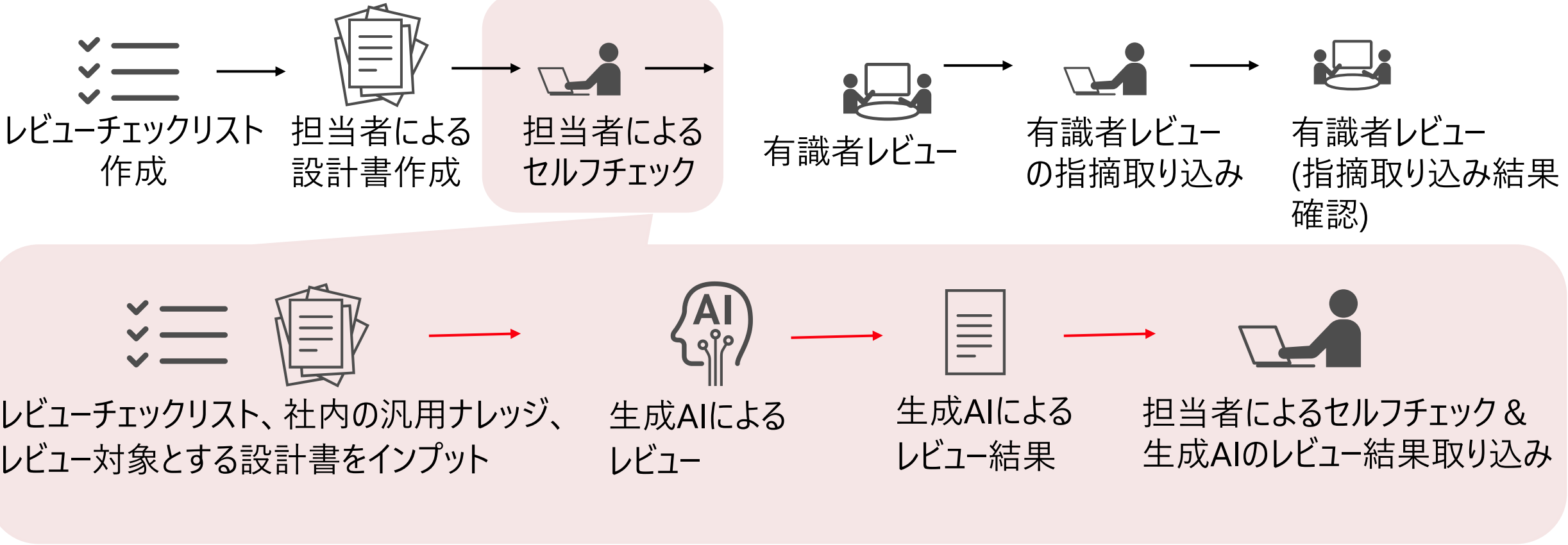
課題3説明（3章：スライド19~25）

3.2 施策の詳細と検証内容

HITACHI

■施策

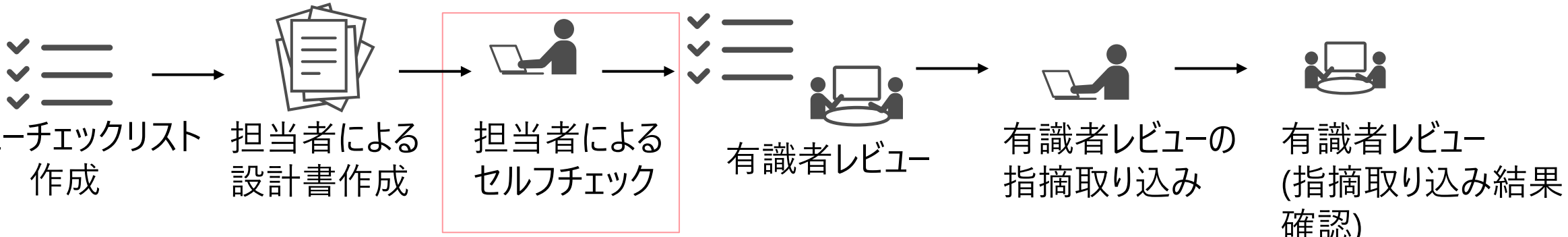
生成AIによって作成したレビューチェックリストのチェック項目文をプロンプトに記載し、生成AIにプロンプトの内容が設計書に含まれているかを確認させる



3.2 施策の詳細と検証内容

HITACHI

	課題	検証内容
課題3	 レビュー成果物と レビューチェックリストの数が膨大 ⇒記載箇所特定/妥当性評価に 工数が膨大	**①工数比較（時間削減効果）** 手作業で実施する際と生成AIを用いる場合の作業工数を比較する。  によるセルフチェック ＆出力確認 VS  によるセルフチェック 【目標】作業時間：50％減 **②生成AIによるレビュー品質の検証** 人の手でセルフチェックした際に摘出されている不良が実際に生成AIでセルフチェックした場合に摘出できているかを検証する。 【目標】正答率：70％以上



3.3 有効性検証結果と評価（作業時間の削減効果）

HITACHI

Point

生成AIを利用時は手作業に比べ作業時間が45分以上削減でき、比率にすると約54%作業時間を削減

#	比較条件	レビュー時間	評価
1	手作業	2h15min	−
2	生成AI利用	1h02min	○

[凡例(評価欄)]　◎目標の倍以上、○目標どおり(作業時間半減)、×目標を下回る(仮説で定義した数値の半分以下)、−対象外

考察

✓ 良かった点

手作業で実施するよりも生成AIを利用した場合の方が、レビュー時間は短い

- レビュー対象の物量が多い、セルフチェックリストが膨大である場合は生成AIを用いることで、さらに作業時間の削減効果が期待できる可能性がある

× 問題点

NG箇所の起票や修正については手作業での作業があり、さらに生成AIの出力結果の確認があるため、全自動にはならない

3.3 有効性検証結果と評価（生成AIによるレビュー品質の検証）

HITACHI

Point

摘出した不良の件数は、手作業と同等の水準

項目	対象CL件数	正答数	正答率	評価※
仮説	21	15	70%	−
実績	21	16	76%	○

※ [凡例(評価欄)]　◎目標の倍以上、○目標どおり、×目標を下回る(仮説で定義した数値の半分以下)、−対象外

考察

良かった点

- 体裁レベルの指摘のみならず**処理漏れなど設計内容に関わる不良を摘出**できた
- 設計書に該当のチェックリスト項目の記載がある場合、記載箇所は生成AIに出力させることでセルフチェックの効率が向上できる

問題点

- 正答率が目標値をわずかに上回るレベルのため効果が薄い
  - インプットする**チェックリストの記載に曖昧表現がある場合、正答率が下がる**
  - 設計書に記載されているか未記載かの2択方式でプロンプトを作成したため、要求仕様ではないことが理由による**設計対象外が全て記載漏れの不良として出力**されてしまう

3.3 有効性検証結果と評価（生成AIによるレビュー品質の検証）

HITACHI

Point

プロンプトの改善により、正答率向上

項目	対象CL件数	正答数	正答率	評価※
改良前プロンプト	21	16	76%	−
改良後プロンプト	21	19	90%	○

※[凡例(評価欄)]　◎目標の倍以上、○目標どおり、×目標を下回る(仮説で定義した数値の半分以下)、−対象外

考察

良かった点

正答率が9割と、比較的正しい内容が返答されているため、人手による訂正や再確認の工数を抑制できると考えられる。

問題点

100％の正答率にはならない
生成AIの出力は人の手で確認する必要がある

知見

■ レビューチェックリストを作成する際は可能な限り項目内容の詳細を明確化する

■ チェックリストの内容が設計書に記載されている/記載されていないの2択ではなく、設計不要（要求仕様ではない）を追加した3項目で生成AIに評価させる

Contents

4 まとめ(1)

	課題	成果
課題1	 レビューチェックリストへのフィードバックは時間がかかる ⇒メンテナンスが追い付かない	生成AIを利用時は手作業に比べ**作業時間が半減**できた
課題2	 暗黙知やプロジェクト独自の観点を含める必要がある ⇒レビューチェックリストの質(スキル依存)	手作業で作成する場合と比べても品質に大きな違いがないことがわかった
課題3	 レビュー成果物とレビューチェックリストの数が膨大 ⇒記載箇所特定/妥当性評価に工数が膨大	生成AIを利用時は手作業に比べ**作業時間を半減**できた 摘出した**不良の件数は、手作業と同等の水準**であることがわかった

今後の課題 インプットデータが大きいプロジェクトやウォーターフォール開発への適用、有効性の検証
⇒ウォーターフォール開発で工程内(序盤)で摘出したバグをレビューチェックリストに取り込み、生成AIにセルフチェックさせることで効率的に品質を担保
AIエージェント化の検討

4 まとめ(2)

HITACHI

■ プロンプトの書き方を工夫し、結果の精度を上げた。

初版

#役割
あなたはソフトウェア開発企業のシニアエンジニアです。品質分析の能力は誰よりも優れています。

#指示
以下の手順に従って、単語リストの単語が含まれる文章を抽出したうえで、要約してください。

#前提
指摘一覧と単語リストがインプット情報として与えられます。

#手順
1.指摘一覧を閲覧し、単語一覧に含まれる単語が含まれる文章をすべて抽出してください。...(略)

修正版

#役割
あなたは文章校閲のプロです。あなたはソフトウェア開発に関する本を出版している部署の人間で、...

#指示
以下の手順に従って、単語リストの単語が含まれる文章を抽出したうえで、要約してください。

#前提
以下の形式でインプット情報が与えられます。
##インプット例
【インプット】
インプット情報①：
インプット情報②：

#手順
1.インプット情報①とインプット情報②を閲覧してください。
1.インプット情報②の単語を上から順に確認します。単語がまだ分析していない単語であればその単語を抽出します。...(略)

①手順に応じた役割を与える今回は要約などの作業のため文章校閲のプロとした

②前提条件や制約条件を記載

③Markdown(インプットも)

④インプットが2種類あるためどちらのインプットから情報抽出/閲覧するかを明示的に記載する。手順を詳細に書き出す(コーディングに似ている)

HITACHI