なぜなぜ分析を“属人的ノウハウ”から“判断基準”へ

～　判断を設計し、生成AIでも再現可能にする　～

チーム２

明石　光介　(株式会社リンクレア)
秋谷　勤　(日本電気株式会社)
梯　雅人　(株式会社日立システムズ)
神崎　和洋　(SCSK株式会社)
島貫　さやか　(株式会社システムインテグレータ)
水野　啓之　(トヨタシステムズ)
横尾　清吉　(株式会社ゼネラル)
陸野　礼子　(株式会社日新システムズ)　(五十音順)

本発表で伝えたいこと

私たちが対象にしたのは、障害分析や不具合分析で実施する「なぜなぜ分析」です
⇒ 同じ事象を分析しても、人によって原因や対策が変わってしまうことがあります

■ **なぜ、なぜなぜ分析は安定しないのか？**

- 手順の問題ではなく、
  分析中の「判断の揺れ」に原因があるのではないかと考えた

■ **今回の提案**

👉 判断を設計する (3つ)

① **観点**：どこを見るか？
② **分岐**：どこで原因を分けるか？
③ **深さ**：どこまで掘るか？

■ **本取組みのポイント**

👉 **「どこを見るか・どこで分けるか・どこで止めるか」**を設計する
この判断基準を**生成AIでも再現可能な形**にした

前期(16期)試行を起点に、今期(17期)考えたこと

前期(16期)の試行結果

- 生成AIに「手順」を指示し、対話形式でなぜなぜ分析を支援させる試行を行った
- しかし、分析の方向づけは安定せず、**経験者の介入・方向修正が必要だった**

今期(17期)の取組み

- 前期は手順を与えたが、分析は安定しなかった
- そこで私たちは、手順ではなく分析中の**「判断」そのものが揺れている**のではないかと考えた
- 特に「どこを見るか」「どこで分けるか」「どこまで掘るか」という判断に着目した
- その判断を、生成AIでも再現できる**「判断基準」**として設計することを目指した

現場で起きていた3つの判断の揺れ

現場で起きていたのは、「なぜを聞く回数」の問題というより、
判断が人に依存して揺れている状態だった

① どの観点で考えるか　　＝ 観点の揺れ

- 作り込みだけを深掘りし、レビューやテストの観点が後追いになる
- 管理や仕組みの観点まで見きれない

② どこで原因を分けるか　　＝ 分岐の揺れ

- 複数の原因を1つの流れとして扱ってしまう
- どこまでを1つの原因と見るかが人によって違う

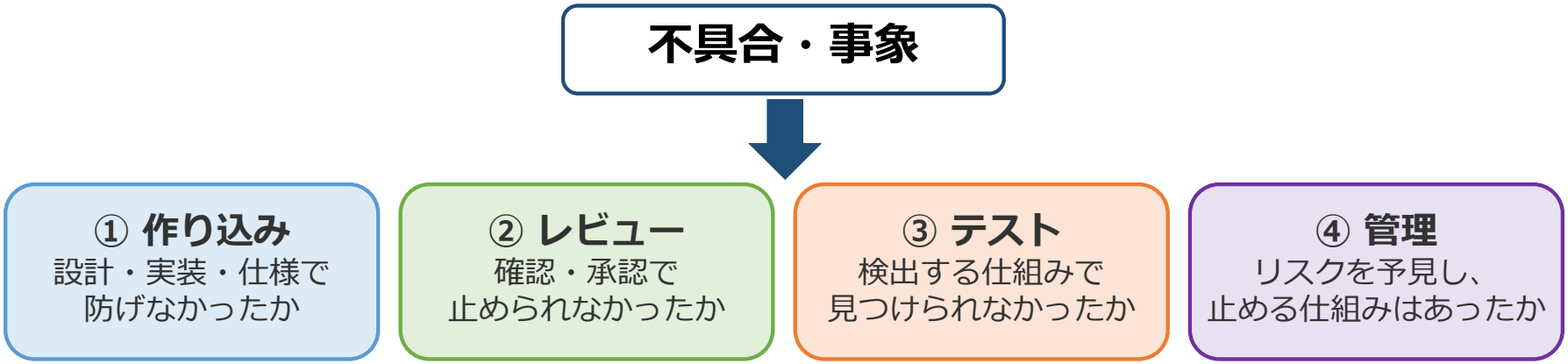
③ どこまで深掘りするか　　＝ 深さの揺れ

- 「まだ浅い」と言われる
- 一方で「そこまで掘る必要があるのか」とも言われる

👉 本取組みでは、この3つの揺れを安定させるために、
「観点」「分岐」「深さ」の判断基準を設計した

判断基準①：どの観点で考えるか

深掘りに入る前に、**必ず4つの観点を確認**



確認する問い：この観点で「本来、どこで止められたか？」

どれか1つの観点だけで終わらせない

※ 三層防御・管理防御の考え方を、観点を揃えるための判断基準として活用(詳細はAppendixを参照)

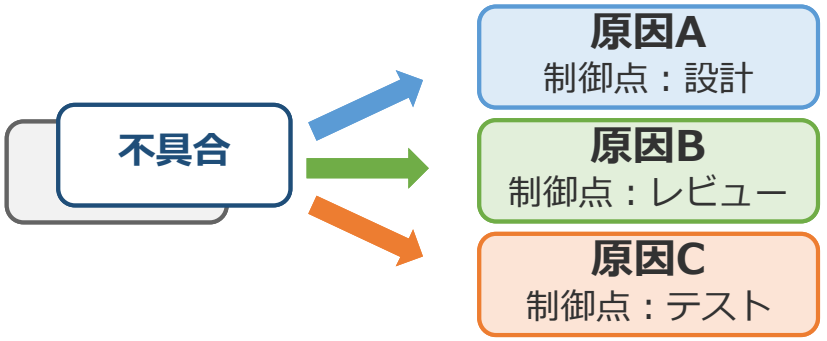
判断基準②：どこで原因を分けるか

原因を“流れ”ではなく、
「**制御点** = 不具合が本来ここで止まるべきだった工程・チェックポイント」で分離

これまで：1つの流れとして扱う

どこで分けるかが人によって揺れる
= 属人的

今回：制御点の違いで別分岐にする



制御点が違う⇒ 別分岐
制御点が同じ ⇒ 同一分岐

制御点 = 本来止めるべき場所

※ 欠陥モデリングの考え方を、分岐を安定させるための判断基準として活用(詳細はAppendixを参照)

観点・分岐を揃えても、まだ残った問題

観点・分岐を揃えることで、見落としは減らせた
⇒ 分岐を明確にすることで、分析対象の曖昧さも減らせた

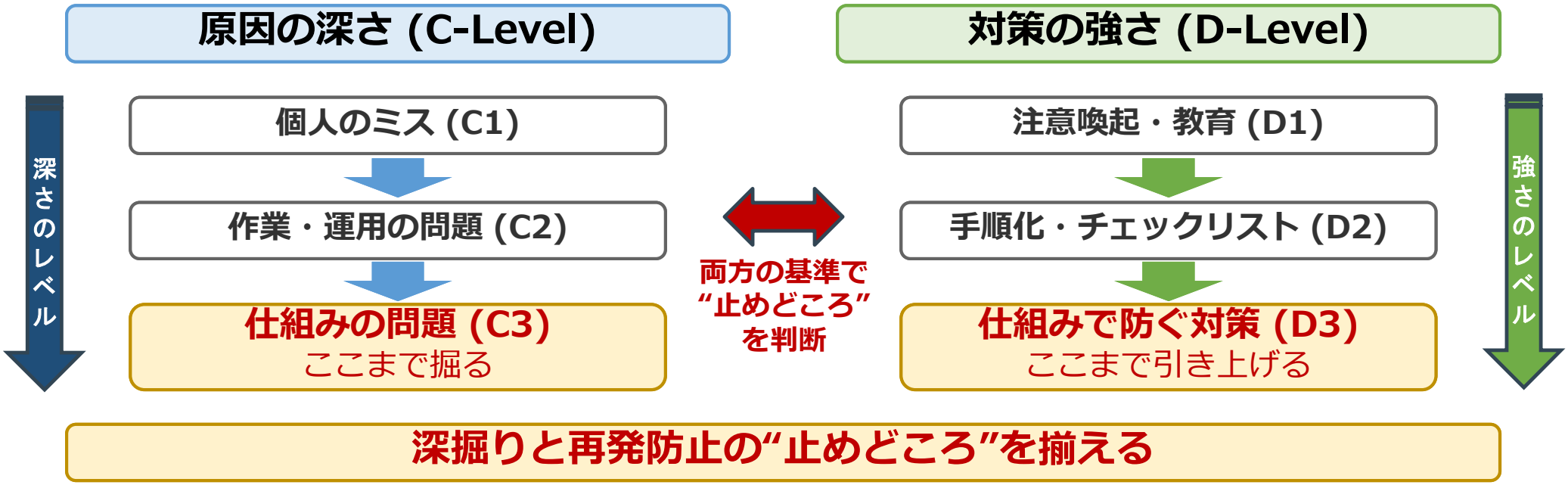
- しかし、それでも次の判断は残った
  - ✓ この原因は、もう根本原因と言えるのか？
  - ✓ ここで深掘りを止めてよいのか？
  - ✓ この対策は、再発防止と言えるのか？

👉 つまり、**「観点」「分岐」と「深さ」は別の問題**だった

そこで、どこまで深掘りすれば十分かを判断する基準を設計した

判断基準③：どこまで深掘りするか

深掘りを個人のミスで止めず、原因を仕組みまで掘り、対策を仕組みで防ぐための物差し
⇒ **仕組みまで掘る「原因の深さ」**と、**仕組みで防ぐ対策へ引き上げる「対策の強さ」**の二軸で判断



※ 「原因の深さ」を原因到達レベル(C-Level)、「対策の強さ」を対策設計レベル(D-Level)の二軸で整理
(詳細はAppendixを参照)

ここまでの整理：3つの判断基準

■ 本取組みで設計した判断基準は3つ

観点で迷わせず、
分岐で混ぜさせず、
深さで揉めさせない

① 観点で迷わせない
[判断基準①]
どの観点を観るかを揃える

② 分岐で混ぜさせない
[判断基準②]
どこで原因を分けるかを揃える

③ 深さで揉めさせない
[判断基準③]
どこまで掘るかを揃える

👉 **なぜなぜ分析を「考え方」ではなく「判断の設計」として扱った**

判断基準を生成AIに渡して、何ができるようになったか

生成AIに原因を自由に考えさせるのではなく、判断基準に沿って問いを進めさせた

■ その結果、次の判断を維持できることを確認した

- 観点が偏る ⇒ **作り込み／レビュー／テスト／管理を確認**
- 新たな原因候補が出る ⇒ **制御点の違いで分岐を判断**
- まだ浅いか、十分か迷う ⇒ **原因の深さで止めどころを判断**
- 対策が弱い ⇒ **仕組みで防ぐ対策へ引き上げ**

不安定さの正体は、判断そのものではなく、
人によって判断が暗黙で揺れていたことにあった

👉 そこで私たちは、
「見る・分ける・止める」判断を明示化し、
生成AIでも扱える再現可能な判断基準として設計した

判断基準をプロンプトに組み込んだ例(抜粋)

※ 詳細を読む必要はありません

 # 絶対原則
・1ステップ1質問を厳守
・同時に複数分岐を深掘りしない
・現在分岐が完了するまで次分岐を深掘りしない

 # 防御層評価(制御配置構造)
三層防御
・作り込み：設計／実装
・レビュー：確認／承認
・テスト：検出
管理防御
・既知リスク／過去同様障害／予見可能だったが未防止
・品質＜納期／コストの優先
・逸脱しても工程停止なし

① 観点で迷わせない
＝三層防御・管理防御

 # 欠陥モデル(分岐判定構造)
欠陥モデル＝欠陥内容、混入点、流出点、
顕在化条件、制御点、責任構造の組

 # 制御点モデル(停止起点構造)
・どこで止めるべきだったか
・制御点の有無
・ある場合：なぜ機能しなかったか
・ない場合：なぜ標準／ゲート／責任構造に組み込まれていなかったか

② 分岐で混ぜさせない
＝制御点(欠陥モデリング)

 # 二軸モデル(中核構造)
##原因到達レベル(C-Level)
・C3：プロセス構造
・C4：組織構造／意思決定
・C5：アーキテクチャ／事業構造
→C3未満で分岐終了禁止
##対策設計レベル(D-Level)
・D3：プロセス変更／ゲート追加／自動化
・D4：組織／責任／評価の変更
・D5：アーキテクチャ変更
→ 再発防止策はD3以上を最低1つ含む

③ 深さで揉めさせない
＝二軸モデル

 # 5Whys深掘り
各分岐を1問ずつ5Whysで深掘りする
→ C3以上に到達するまで終了しない

5Whys

 # 新規分岐条件
・欠陥内容／混入点／流出点／制御点／責任構造のいずれかが異なる
・別の三層防御または管理防御の制御点を持つ
・独立して事実確認・深掘りが可能
 # 分岐管理
・独立した原因／欠陥モデル／制御点候補は新規分岐
・分岐は削除しない／同時進行しない
・現在分岐が終了前確認または完了するまで次分岐へ進まない
 # 分岐終了ゲート
・C3以上に到達
・D3以上の対策設計が可能な制御点を特定済
・三層防御または管理防御の制御点を特定し、全て評価済
・未分離の欠陥モデル／制御点が残っていない

分岐管理

 # 全分岐完了後の最終出力
・根本原因一覧(C3以上、影響度(S)/頻度(O)/検出(D)付き)
・欠陥モデル一覧
・なぜなぜ構造要約(ユーザー回答ベース)
・再発防止策一覧(D-Level付き)

 # 基本進行フロー
1. 初期情報収集
2. 分岐作成(三層＋管理)、分岐評価
3. 欠陥モデル整理
4.分岐一覧作成、分岐網羅確認
5. 5Whys深掘り(1問ずつ)
6. ユーザー承認後、分岐完了
7. 全分岐完了後の網羅確認
8. 最終出力

試行事例：うるう日による免許作成システム障害

公開情報をもとに、「うるう日」を契機とした免許作成システム障害を分析演習として試行

※ 公開情報をもとにした分析演習として整理したものであり、実際の関係組織の公式見解ではありません

■事例のポイント

1.不具合概要

- 発生日時：2024年2月29日(うるう日) 午前8:00～
- 影響地域：愛媛県、新潟県、神奈川県、岡山県(4県警)
- 影響内容：免許証の即日交付が不可、約1,000人規模で交付遅延・後日対応が発生

2.直接原因候補 ⇒ 分析演習で扱う原因候補

- 免許作成システムの日付管理に不備があり、うるう日を正しく扱えなかった可能性がある
- 過去類似問題の教訓が、設計・レビュー・テストに反映されていなかった可能性がある

3.分析で確認したい論点

- 設計で防げたか
- レビュー／テストで見つけられたか
- 過去類似問題の教訓が反映されていたか

参照元：

・読売新聞オンライン 「うるう日」設定ない機器がエラー、４県警で免許証作成でシステム障害…スギ薬局も一時精算できず - 読売新聞

・テレ朝news 運転免許作成システム障害「うるう年」が原因　新潟の一部除き復旧済み

・新潟ニュース 原因は“うるう日”!- 運転免許センター システム障害で“運転免許証”発行できず【新潟】 - 新潟ニュース NST

試行結果①：深掘り前に観点と分岐を作成

利用環境：Microsoft 365 Copilot Chat
※ **以降の実行結果は公開情報をもとにした分析演習結果であり、実際の関係組織・メーカーの公式原因分析ではありません**　モデル選択：GPT-5.5 Think deeper(画面表示名)

見るポイント：生成AIはいきなり原因を深掘りせず、4つの観点で分岐を作ってから、深掘り対象を1つに絞った

初期情報が不足しているため、**分岐作成・深掘りはまだ開始しません**。発生日時/対象機能/期待結果と実結果/影響/検出タイミング等を教えてください

4県警の運転免許センターで2月29日に免許証交付処理が不可。共通要素は同一メーカー製の免許作成機・システム。検出は本番稼働開始直後の業務異常

確定事実／可能性情報／未確認事項に分類。さらに**「この欠陥は本来どこで止めるべきだったか？」を基準に、初期分岐を登録します**

■ 生成AIの動き：

【生成AIの動き】
- 深掘り前に観点を揃える
- **「原因」ではなく「止めるべき制御点」で分岐**
- 生成AIが分岐一覧を維持し、**次に掘る対象を一つに絞る**

【ここでのポイント】
利用者が一気に答えても、**生成AIは「事実」「可能性」「未確認」を分けて**扱い、推測で原因確定していない
⇒ プロンプトの絶対ルールが効いている

- 生成AIが初期情報から作成した分岐リスト

分岐ID	テーマ	三層分類	状態	C-Level	管理影響	構造タイプ	備考
B1	うるう日を正常処理できる作り込み制御	**作り込み**	**深掘り中**	未到達	未確認	未確定	最初に深掘り
B2	日付境界・うるう日観点のレビュー制御	**レビュー**	未着手	未到達	未確認	未確定	B1完了後に評価
B3	2/29を含む発行・連携テスト制御	**テスト**	未着手	未到達	未確認	未確定	B1完了後に評価
B4	複数県共通製品の管理防御・横展開制御	**管理防御**	未着手	未到達	あり得る	未確定	管理防御候補

試行結果②：直接原因で止めず、仕組みまで深掘り

利用環境：Microsoft 365 Copilot Chat
モデル選択：GPT-5.5 Think deeper(画面表示名)

見るポイント：生成AIは「日付処理ミス」で止めず、なぜ制御点が機能しなかったかを問い、仕組みの問題まで深掘り

和暦対応の日付処理について、うるう年2月29日を正常日付として扱う仕様・設計標準・実装ルールは存在していましたか？

存在していたが、内容が不十分で**2月29日をカバーしていなかった**。また、**過去に同様問題が発生**していたが、教訓が活かせなかった

過去障害の教訓未反映は、**現分岐に吸収せずB5として登録**します。ただし、**現在はB1の深掘りを継続**します
2月29日が仕様・設計標準から漏れたのは、うるう日などの特殊条件を要件として抽出する観点や手順が不足していたためですか？

例外的な特殊条件を明確にせず設計を進め、外部仕様からも抜け落ちた。仕様策定時の特殊条件などの要件抽出プロセスに漏れがあった

特殊条件が未抽出・未記載のまま外部仕様や設計へ進むことを止める承認ゲート／停止条件／責任役割は存在していましたか？

承認ゲートも停止条件も存在しているが、今回の特殊条件の観点が抜けていたため、必要性のチェックができなかった

B1は達成基準(C3)に到達。構造タイプは標準形骸化型。承認ゲートは存在したが、うるう日・日付処理・例外日付条件の判定観点が含まれていなかった

■ 生成AIの動き：

【生成AIの深掘りが上手く進んだ点】
① 直接原因で停止しない
② **「制御点がなぜ機能しなかったか」を聞く**
③ **途中で出た「過去障害」はB5に分離**
④ 規定基準(C3)到達後も、終了前に人間承認を求める

【C-Levelの推移】
C1：日付計算・例外処理の設計漏れ
C2：仕様・設計標準が2/29をカバーせず
C3：特殊条件を要件抽出するプロセス／承認ゲート観点の不足

試行結果③：レビューとテストを混ぜずに分離

利用環境：Microsoft 365 Copilot Chat
モデル選択：GPT-5.5 Think deeper(画面表示名)

見るポイント：生成AIは「見逃し」と一括りにせず、レビューとテストを止める仕組みの違いで別分岐にした

B2：レビュー制御

レビューは存在していたが、特殊条件確認や影響調査が網羅的に行えているかの仕組みがなく、確認が属人的になった

判定基準や証跡要求が全くなかったわけではなく、**部分的には存在**していた。問題は網羅性にあった

B2は 達成基準(C3)に到達。レビュー基準更新時に、未知・未登録の特殊条件を新たに洗い出す入力情報や手順が不足していた

B3：テスト制御

テストも設計ベースで行ったため、特殊条件を含めたテスト項目がなかった

テスト設計標準に、設計書外の特殊条件・境界条件を**独立抽出する必須ルールはなかった**

B3は達成基準(C3)に到達。設計書外条件を独立抽出し、未設定時に止めるテスト制御の強制力不足

■ 生成AIの動き：

【ここで見せる生成AIの働き】
レビューとテストを「見逃し」と一括りにせず、
・レビュー　：仕様／設計の不足を検出する制御
・テスト　　：設計書外条件を独立抽出する制御
として別々に扱う

【到達結果】
B2(レビュー)　　：C3／標準形骸化型
B3(テスト)　　：C3／強制力欠落型
⇒ **分岐ごとにD3以上の対策設計が可能な制御点を特定**

試行結果④：全分岐完了後にD3以上の対策へ接続

利用環境：Microsoft 365 Copilot Chat
モデル選択：GPT-5.5 Think deeper(画面表示名)

見るポイント：生成AIは全分岐の確認後に、注意喚起ではなく、仕組みで防ぐ対策へ接続した

複数県で同一メーカー製システムに同時発生。横展開責任・品質ゲートが独立制御点になり得ます

同時に複数県で発生しているため、**横展開の議論ではない**です。B4は未成立で良いです

B4は未成立として整理します。次に、過去障害の教訓を標準・ゲートへ反映する制御(B5)を確認します

再発防止管理は存在したが、完了条件が「分析完了」中心で、標準・レビュー・テスト・ゲートへの実反映完了を含んでいなかった

B5は達成基準(C3)に到達。過去障害の教訓を実反映せずとも完了扱いにできていた。強制力欠落型として整理。**全分岐完了後に対策を提示します**

■ 生成AIの最終出力結果：

【最終的な根本原因(要約)】 ※ **生成AI対話で整理された根本原因候補**

- 特殊条件を要件化する**作り込み**ゲートの観点不足(C3)
- 未知・未登録条件を洗い出す**レビュー**入力／手順不足(C3)
- 設計書外条件を独立抽出し、未設定時に止める**テスト**制御不足(C3)
- **管理**/教訓を実反映しなくても再発防止完了扱いにできる完了条件(C3)

【対策案(D3以上)】 ※ 根本原因から生成AIが抽出した対策候補

- [作り込み]特殊条件要件抽出ゲートの追加(D3)
- [レビュー]レビュー基準更新時の未知条件洗い出しプロセス追加(D3)
- [テスト]設計書外条件の独立抽出テストゲート追加(D3)
- [テスト]うるう日／和暦変換の回帰テスト自動化(D3)
- [管理]再発防止完了条件を「実反映完了」まで拡張(D3)
- [管理]再発防止策の完了追跡KPI化(D4)

【この試行で確認できたこと】
生成AIが優れていたのは、原因を当てたことではない。**「見る・分ける・止める」という判断基準を生成AIが守り続けたこと**

チームメンバ試行で得られた示唆

■ **確認できたこと**

- 人対人では踏み込みにくいところまで深掘りできた
- **場の空気や発言力に左右されず**、判断基準に沿って深掘りの方向性を保てた
- **社内分析に近い内容が網羅された**
- 設計／レビュー／テスト観点では高い一致が見られた

■ **見えてきた課題**

- 顧客対応や運用影響など、**業務文脈は人の補完が必要**
- より具体的な対策には、過去事例や業務特性の組込みが有効

👉 **判断基準により分析構造は安定するが、現場文脈を加えることで実務適用性が更に高まる**

まとめ：判断を設計することで再現可能にする

■ **結論**

- なぜなぜ分析の不安定さの正体は、**判断が暗黙で、人によって揺れていた**ことにあった
- 本取組みでは、3つの判断を基準化した
  - ① **観点**：どこを見るか
  - ② **分岐**：どこで分けるか
  - ③ **深さ**：どこで止めるか

👉 私たちは、なぜなぜ分析の**手順を改善したのではありません**
分析者が暗黙に行っていた「どこを見るか・どこで分けるか・どこで止めるか」という
判断を明示化し、再現可能な判断基準として設計しました
生成AIは、その判断基準を一貫して適用できることを確認しました

■ **最後に**

なぜなぜ分析の本質は、**考え方ではなく、判断基準を設計すること**にあると考えます

まずは、**“観点を並べ、原因を分け、仕組みレベルで止める”**ことだけでも試してみてください
それだけでも、なぜなぜ分析の揺れを減らす第一歩になるはずです

※ ご関心があれば、今回使用したプロンプト構造(そのまま再現可能)も紹介できます

ご清聴ありがとうございました

※ 以降はAppendix

Appendix：補足・参照用スライド

観点を揃えるために活用した考え方（三層防御・管理防御）

※ 本編「判断基準①：どの観点で考えるか」の補足

■ **「深掘りの前段」で、必ず4つの防御層(観点)をチェック**

- 作る段階で防げなかったか？(設計・実装・仕様) ⇒ 「作り込み」の観点
- 確認・レビューで止められなかったか？ ⇒ 「レビュー」の観点
- テスト・検出の仕組みは十分だったか？ ⇒ 「テスト漏れ」の観点

三層防御の観点

- リスクは予想できたはずなのに、なぜ止められなかったか？⇒ 「管理防御」の観点

(個人ではなく「構造」を見る観点、三層防御の上位視点)

👉 **どれか一つの観点だけでは終わらせない**

⇒ 三層防御・管理防御の考え方を判断基準に採用!!

[ご参考]

この観点は、従来の三層防御・管理防御として事故分析や品質分野で使われてきた考え方です

- 三層防御　：作り込み／レビュー／テスト漏れで「防げたはずの層」を見る考え方
- 管理防御　：それらがなぜ機能しなかったかを「意思決定・評価構造」まで遡って見る観点

※ 本発表では三層防御・管理防御を「原因分類」ではなく、
「個々の深掘りに入ってよいかを判断する基準」として使用しています

深掘りに入る前の判断用チェックポイント

分岐を安定させるために活用した考え方（欠陥モデリング・制御点）

※ 本編「判断基準②：どこで原因を分けるか」の補足

■ **分岐ルールの考え方**
- 分岐の判断を「人の解釈」ではなく、構造として定義

■ **欠陥モデルを構成する要素で整理**
- 欠陥内容
- 混入点(どこで入ったか)
- 流出点(どこで見逃したか)
- 制御点(どこで止まるべきだったか)
- 責任構造

■ **制御点を分岐の主軸に採用**
- 「なぜ起きたか」ではなく、**「なぜ止まらなかったか」を起点にする**

◆従来の状態(問題)◆
原因A → 原因B → 原因C
(本来は別問題なのに)
- 1つの流れとして扱われる
- どこで分けるかが人依存

◆今回の考え方(構造化)◆
不具合 { 原因A(制御点①) / 原因B(制御点②) }
- 制御点が異なる ⇒ 別分岐
- 各分岐を独立して深掘り

◎分岐ルール(最重要)◎
- 制御点が異なる ⇒ 新規分岐
- 同一制御点 ⇒ 同一分岐

※ **実際には欠陥モデル全体(欠陥内容・混入点・流出点・責任構造)で判定**しているが、本発表では「制御点」を代表軸として説明

👉 **『原因の連鎖(因果)』ではなく、『構造』として分離することで、分析単位を安定化**

※ 本発表では欠陥モデリングを一般的な原因の整理目的ではなく、「分岐を安定させるための構造」として使用しています

欠陥モデリングから本手法への再構成

※ 本編「判断基準②：どこで原因を分けるか」の補足

■ 欠陥モデリングとの関係

- 欠陥モデリングは、原因を構造的に分解し、深掘り対象（誘発・過失・増幅）を抽出することで、**「なぜ起きたか」**という因果連鎖を分析し、再発防止および未然防止に繋げるためのモデル
- 本手法ではその考え方を参考にしながら、**「どこで生じ、どこで止められたか」**に着目して整理

【欠陥モデリング(Fabre)】

[誘発因子] → [過失因子] → [欠陥] → [表出現象]

[増幅因子]

▶ 因果連鎖(**なぜ起きたか**)

再構成

【本手法(分岐構造)】

[混入点] → [欠陥内容] → [流出点] → [表出現象]

[制御点] **[制御点]** **[制御点]**

[責任構造] [責任構造] [責任構造]

▶ 制御構造(**どこで止めるべきだったか**)

【本手法での捉え方(視点を変えて整理)】 ※ 原因を説明する視点から、防御を分析する視点へ変更

Fabreの分析視点	本手法の分析視点	目的
誘発因子・過失因子	混入点・責任構造	なぜ起きたか → どこで生じたか
欠陥	欠陥内容	欠陥そのものを表現
増幅因子	流出点	なぜ拡大したか → どこで見逃したか
(本手法で主軸化)	**制御点**	**本来どこで止めるべきだったか**

👉 **本手法では“制御点”を主軸とし、「どこで止めるべきだったか」を明確化**

深掘り／再発防止のための判断基準 (C-Level／D-Level)

※ 本編「判断基準③：どこまで深掘りするか」の詳細

■ 判断基準としての二軸モデルを提案

① どこまで来たら「根本原因」と言えるか？
[原因到達レベル] 深掘りの十分さを規定

原因到達レベル (C-Level)	定義
C1	行動・作業
C2	工程・運用
C3	プロセス構造
C4	組織構造・意思決定
C5	アーキテクチャ・事業構造

⇒ 原則：C3未満での終了禁止

② それは「再発防止」と言えるか？
[対策設計レベル] 再発防止策の十分さを規定

対策設計レベル (D-Level)	定義
D1	注意喚起・教育
D2	手順明文化・チェックリスト
D3	プロセス構造変更(ゲート追加・自動化等)
D4	組織構造変更(責任構造・評価制度変更)
D5	アーキテクチャ変更

⇒ 原則：D3以上を最低1つ含めること

※ C3/D3を基準にしたのは、人が「再発防止として説明できる」と合意できた、最低限の構造単位を経験的に置いたものです

[補足] 上記規定は**一般的な標準や成熟度モデルではありません**
人がレビュー・合意するために、以下の**止めどころの判断を揃える目的で設計した、本発表独自の判断基準**です
- どこまで来たら、「根本原因」と言えるか？
- どこまで具体化したら、「再発防止策」と言えるか？

※ 二軸モデルの詳細は次スライド参照

二軸モデル(1/4)：位置付け

※ 本編「判断基準③：どこまで深掘りするか」の詳細

■ 従来の考え方との関係

- 本発表の二軸モデルは、
  「人は必ず間違える」という人間工学・安全工学の既知の事実に基づき、
  個人の意識ではなく、**「組織・プロセス構造」を改善するための判断指標**です
- 既存の理論・標準を置き換えるものではなく、
  なぜなぜ分析の「止めどころ判断」に特化して再整理したものです

二軸モデル(2/4)：原因到達レベル(C-Level)

※ 本編「判断基準③：どこまで深掘りするか」の詳細

■ **「なぜ」がどの深さまで到達したか**
トラブルの真因が「個人のミス」にあるのか、それとも「仕組みの欠陥」にあるのかを判定します

原因到達レベル (C-Level)	定義	背景にある考え方
C1	行動・作業	ヒューマンエラー指摘(誰が何を間違えたか)
C2	工程・運用	工程不備・運用逸脱の分析
C3	プロセス構造	なぜなぜ分析の原則／スイスチーズモデルの「構造欠陥」
C4	組織構造・意思決定	管理防御・マネジメント要因
C5	アーキテクチャ・事業構造	事業・経営構造による制約

【鉄則】C3未満で終了禁止

- トヨタ生産方式の「なぜなぜ分析」や、
ジェームズ・リーズンのスイスチーズモデルが示す通り、
真の原因は常に個人の背後にある「プロセスや構造(C3以上)」に存在
- ここを特定しない限り、人を替えて必ず再発します

二軸モデル(3/4)：対策設計レベル(D-Level)

※ 本編「判断基準③：どこまで深掘りするか」の詳細

■ **「再発防止策」がどこに作用するか**
対策が「人の注意」に頼るものか、それとも「物理的に防ぐ」ものかを判定します

対策設計レベル (D-Level)	定義	背景にある考え方
D1	注意喚起・教育	人の記憶や集中力に依存する弱い対策
D2	手順明文化・チェックリスト	遵守前提のソフト対策
D3	プロセス構造変更(ゲート追加・自動化等)	Hierarchy of Controls／ポカヨケの考え方
D4	組織構造変更(責任構造・評価制度変更)	責任・評価構造による統制
D5	アーキテクチャ変更	設計・前提条件の刷新

【鉄則】D3以上を最低1つ必須

- 安全工学の「対策の階層(Hierarchy of Controls)」や、
製造現場の「ポカヨケ(フールプルーフ)」が示す通り、
教育や注意喚起だけでは有効性が低い
- 人の記憶や集中力に依存しない「物理・構造的な仕組み(D3以上)」を組み込んで、
初めて実効性のある再発防止となります

二軸モデル(4/4)：まとめ(再掲)

※ 本編「判断基準③：どこまで深掘りするか」の詳細

■ この二軸モデルの目的は
- 経験則に頼っていたトラブル対応を
- 科学的・構造的な判断へ転換すること

1. C3以上で「個人の背後にある構造的弱点」を特定し
2. D3以上で「誰がやっても失敗しない仕組み」を作る

👉 **この一貫性こそが、組織全体の品質を底上げする中核となります**

前期(16期)試行との比較

重要な前提：前期(16期)では、GPT-4系を使い、
- 生成AIになぜなぜ分析を手伝わせるため、
- 「考え方」や「対話手順」を中心に設計(今回の取組み①②③はなし)

したが、分析の深さや止めどころは安定せず ⇒ **[結論] 経験者の介入が必要不可欠**

■ **前期(16期)：前期の試行(GPT-4系)では、**
- 観点・分岐・深さ・停止条件の維持が安定しなかった
- 「止める／続ける」という判断を安定して行えない
- 判断の前提や制約をプロンプトに埋め込んでも、思考の途中で揺れや無視が起きた

■ **今期(17期)：今期の試行(GPT-5系)で確認できた変化**
- 観点・分岐・深さ・停止条件を同時に守れた
- 「止める／続ける」という判断を生成AIが判断基準に沿って支援できた
- 明示的に設計した判断基準を、思考の途中でも崩さず保持できた

👉 **今期の試行では、判断基準を途中で崩さず保持しながら対話を継続できるケースが確認できた**