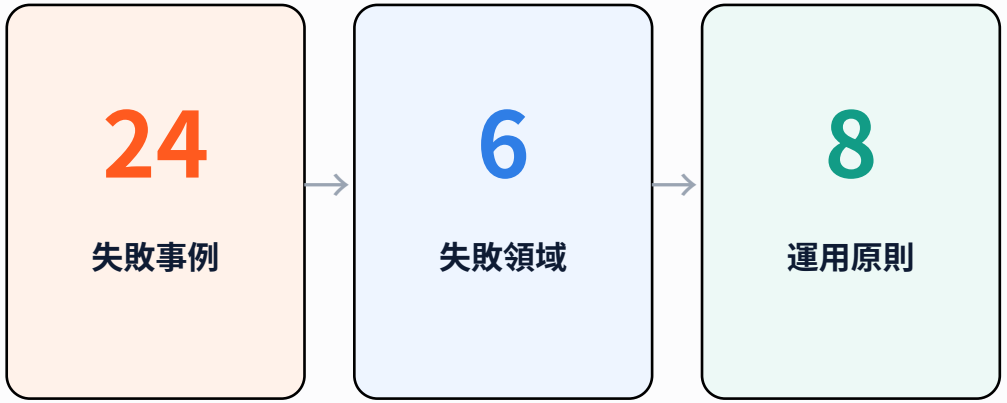
AI エージェント運用の失敗学

24 の失敗事例から、次の運用ルールをつくる。

Netsujo で実際に起きた 24 事例を、 6 つの失敗領域に整理。
最後に 8 つの運用原則と、毎回使う確認順へまとめた。



この資料の流れ

何が起きたか → どこで判断を誤ったか → 次に何を変えるか

対象： AI に開発・資料制作・業務を任せる経営者／事業責任者／開発リーダー

失敗事例を、再発防止のルールに変える。

24 の事例を 6 領域に整理し、最後に 8 つの運用原則にまとめる。

STEP 1

事例を読む

何が起きたか・なぜ起きたか

STEP 2

対応を見る

どう直したか・何を残すか

STEP 3

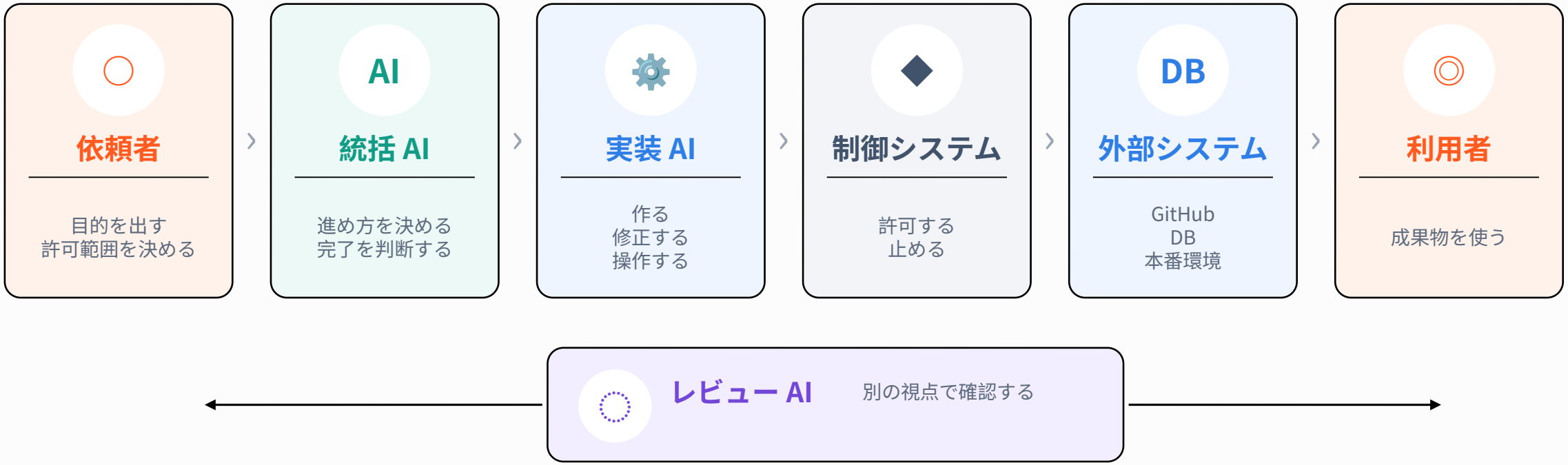
原則にする

次に何を変えるか・再発を防ぐ

目的は反省ではなく、再発防止の設計である。

事例は「誰が判断し、誰が動いたか」で読む。

依頼・判断・実行・確認を分けると、失敗の原因が見えやすくなる。



見る点：誰が依頼し、誰が判断し、誰が実行し、誰が確認したか。

用語は 7 つだけ。これだけ分かれば読める。

用語の意味を固定してから、 24 事例へ進む。

PR
ひとまとまりの変更提案。修正が入れば中身も変わる。

現在の状態
今の版・担当・停止状況。再開前に取り直す。

版（SHA）
その時点の内容を示す識別子。レビューはこの版に固定する。

外部変更
DB ・本番・送信・権限変更など、外部へ残る操作。

CI
自動で走る検査。緑でも必須検査が未実施のことがある。

操作 ID
外部操作を一意に追う番号。結果不明のときは、再送前に照合する。

実行環境
AI やコマンドが実際に動く条件。端末が同じでも版や設定は変わる。

用語は短く固定する。意味が揺れると、判断も揺れる。

24 事例を、 6 つの失敗領域に整理する。

問題を原因別に並べると、次に足すべき運用ルールが見える。

✓ 完了・受入

完了条件が曖昧で終わらない

↻ 状態・再開

状態が残らず再開できない

◎ 担当・委任

誰が判断・実行するか混線する

⚙ 実行環境・制御

権限や接続で実行が止まる

◌ レビュー・証拠

確認と証拠が不足する

▥ 収束・優先順位

論点が散って収束しない

⚑ **まずは領域に分ける。原因を混ぜない。**

01 完了・受入

利用者が目的を達成するまで確認する。工程完了だけで終わらせない。

判断基準 **利用者が目的を達成するまで確認する。工程完了だけで終わらせない。**

典型的な失敗の流れ

01 工程の完了で止める › 02 早く「完了」と報告する › 03 利用者の目的が未達のまま残る

該当ケース CASE 01 CASE 02 CASE 03 CASE 21

⚑ 4 件の事例を、**完了・受入**としてまとめて読む。

本番反映まで終えた。でも、利用確認前に「完了」と報告した。

統括 AI　実装 AI

出典：内部 I1 §1,5,13–14

01 起きたこと

本番反映の完了を、そのまま依頼完了と判定した。

実装✓ / 自動検査✓ / 配備✓ / 利用確認 × / 目的達成 ×

02 判断のズレ

本番反映まで終わった ≠ 利用者の目的も達成した

確認すること： 利用者が目的を達成できたか

03 直したこと

着手時に「誰が何をできれば完了か」を固定。対象版で利用結果を確認してから、統括 AI が完了を判定する。

04 確認結果

設計済み｜効果は未測定

利用確認を完了条件に分ける規則は設計済み。誤完了率はまだ測っていない。

検査は合格。でも、人には読みにくかった。

実装 AI　レビュー AI

出典：会話 C1 / 内部 I3

01 起きたこと

自動検査の PASS を、読みやすさの PASS とみなした。

構文・はみ出しは PASS でも、文字量・改行・読む順番は別問題。

02 判断のズレ

機械検査が通った ≠ 人にも読みやすい

確認すること： 最終画面を実際の表示条件で読んだか

03 直したこと

全ページを画像で確認し、改行・余白・読む順番を修正。レビューは最終表示を見て判定する。

04 確認結果

改善を確認｜追加修正あり

見やすさの改善評価は得た。説明不足が残り、追加修正した。

48 枚から 76 枚へ。それでも「なぜ」が伝わらなかった。

実装 AI

出典：会話 C1

01 起きたこと

説明不足に対し、因果を直さずページ数だけ増やした。

増えた：用語・手順・補足　／　不足したまま：なぜ誤ったか

02 判断のズレ

情報量を増やした　≠　理由まで伝わった

確認すること： 読者が「なぜ失敗したか」を説明できるか

03 直したこと

各事例を「事実 → 判断のズレ → 対策 → 確認結果」で再編集。
補足より先に、因果を主図で示す。

04 確認結果

改善評価あり｜本版は未受入

過去版は改善評価を得た。本版は主語と因果を再修正中。

一度ログインできた。それだけで、招待機能の修正を終えた。

実装 AI　レビュー AI

出典：内部 I1 §6,13–14

01 起きたこと

一度ログインできただけで、継続利用まで直ったと判断した。

未確認：再読込 / 再ログイン / 期限切れ / 別組織拒否

02 判断のズレ

一度ログインできた ≠ その後も正しく使える

確認すること： 再ログイン・期限・所属・権限まで確認したか

03 直したこと

状態の変化ごとに、条件・操作・期待結果を固定。検証用利用者で再ログイン後の所属・権限・保存まで確認する。

04 確認結果

設計済み｜実製品は未測定

再ログイン・期限・所属・権限まで確認する試験を設計。実製品での効果は未確認。

02 状態・再開

再開前に現在の状態を取り直す。会話や前回記録は参考情報として使う。

判断基準 **再開前に現在の状態を取り直す。会話や前回記録は参考情報として使う。**

典型的な失敗の流れ

01 前回記録をそのまま信じる
›
02 古い状態のまま再開する
›
03 二重作業・誤操作が起きる

該当ケース CASE 04 CASE 05 CASE 14

⚑ 3 件の事例を、状態・再開としてまとめて読む。

会話が消え、次の担当は「途中の状態」を読めなかった。

統括 AI　実装 AI

出典：内部 I1 §7.4,10–11

01 起きたこと

会話内の記憶を、再開用の記録として扱った。

不足：目的 / 対象版 / 担当 / 残件 / 未確定操作

02 判断のズレ

会話に残っている ≠ **次担当が再開できる**

確認すること： 別セッションから必要情報を読めるか

03 直したこと

目的・対象版・担当・残件・結果不明操作を永続記録へ保存。
再開時は現在の状態と照合する。

04 確認結果

設計済み｜復元率は未測定

再開記録の項目と継承方法は設計済み。セッション消失後の復元率は未測定。

「続けて」で、古い作業状態から再開した。

実装 AI　統括 AI

出典：内部 I1 §10–11 / GitHub G2

01 起きたこと

前回記録を読み直しただけで、現在の状態を再取得しなかった。

前回： H0 / 担当 A / 通常作業　→　現在： H1 / 担当 B / 停止中

02 判断のズレ

最後の記録が正しい　≠　現在も同じ状態

確認すること： 対象版・担当・停止・本番反映結果を再取得したか

03 直したこと

再開前に対象版・検査・担当・停止・本番反映結果を取り直す。記録との差分から残作業を更新する。

04 確認結果

当時の不一致を確認

記録した版と実際の対象版がずれた事例を確認。現行システムは今回未監査。

成功応答が消えただけなのに、同じ操作をもう一度送った。

実装 AI　統括 AI

出典：内部 I1 §11 / 一次資料 E3

01 起きたこと

応答が消えたため、外部処理も失敗したと決めつけた。

正：結果不明 → 操作 ID を照合　／　誤：失敗とみなして再送

02 判断のズレ

成功応答がない　≠　外部処理も未実行

確認すること： 同じ操作 ID の外部記録があるか

03 直したこと

操作 ID ・対象・送信時刻を残し、再送前に外部記録を照合。
結果不明の間は同じ資源への再操作を止める。

04 確認結果

設計済み｜二重操作率は未測定

結果不明を独立状態として扱う設計は済み。二重操作が減ったかは未測定。

03 担当・委任

担当・委任範囲・共有資源を先に決める。引き継ぎ条件まで残す。

判断基準 **担当・委任範囲・共有資源を先に決める。引き継ぎ条件まで残す。**

典型的な失敗の流れ

01 役割だけ分ける › 02 共有資源・権限が曖昧になる › 03 競合・不要な確認が増える

該当ケース CASE 06 CASE 09 CASE 10

3 件の事例を、担当・委任としてまとめて読む。

2 つの AI が、同じ共有資源へ同時に書いた。

統括 AI　実装 AI

出典：内部 I1 § 10.1,14

01 起きたこと

担当を分けたが、同じ DB や本番環境を同時に更新した。

担当は別でも、共有 DB / 本番環境への書き込み順序が競合。

02 判断のズレ

担当が別 ≠ **変更先も独立**

確認すること： 共有 DB ・本番環境・実行順序まで分離できているか

03 直したこと

外部変更が及ぶ共有資源を先に列挙。同じ資源への書き込み担当を 1 人にし、他 AI は調査・レビューへ回す。

04 確認結果

設計済み｜競合件数は未測定

共有資源ごとに書き込み担当を 1 人にする規則を設計。競合減少は未測定。

1 人の AI が操作できず、仕事全体を止めた。

統括 AI　実装 AI

出典：内部 I1 §11

01 起きたこと

今の AI に実行手段がないことを、組織全体の実行不能と扱った。

代替の承認済み実行者がいるなら、同じ目的と未確定操作を引き継げる。

02 判断のズレ

自分が操作できない ≠ **誰も操作できない**

確認すること： 承認条件を満たす代替実行者がいるか

03 直したこと

能力不足と操作禁止を分ける。承認条件を満たす代替実行者がいれば、同じ目的・予算・未確定操作を引き継ぐ。

04 確認結果

設計済み｜不要停止は未測定

代替実行者へ引き継ぐ条件は仕様化済み。不要停止の減少は未測定。

「任せた」の範囲内なのに、通常判断を毎回依頼者へ戻した。

統括 AI　依頼者

出典：内部 I1 §3–4,8,13

01 起きたこと

すでに委任された通常判断まで、毎回依頼者へ戻した。

通常修正・再検証・担当交代は範囲内。新規費用・契約・不可逆変更だけ戻す。

02 判断のズレ

迷いがある ≠ **依頼者の再承認が必要**

確認すること： 既存の承認範囲を超える変更か

03 直したこと

目的・許可操作・禁止事項・予算を引き継ぐ。新しい費用・契約・不可逆変更など、承認範囲外だけ依頼者へ戻す。

04 確認結果

設計済み｜割込み回数は未測定

委任範囲内は継続し、範囲外だけ依頼者へ戻す規則を設計。割込み削減は未測定。

04 実行環境・制御

外部変更の直前に、環境・権限・経路を確認する。禁止・停止は実操作で強制する。

判断基準 **外部変更の直前に、環境・権限・経路を確認する。禁止・停止は実操作で強制する。**

典型的な失敗の流れ

01 起動時だけ確認する › 02 実行直前の変化を見逃す › 03 禁止・停止が実操作に効かない

該当ケース

CASE 07 | CASE 08 | CASE 11 | CASE 12 | CASE 13 | CASE 23

⚑ 6 件の事例を、実行環境・制御としてまとめて読む。

停止指示が出た。でも、書き込みは止まらなかった。

制御システム | 実装 AI | 統括 AI

出典：内部 I1 §10.1–10.2

01 起きたこと

起動時の許可を、書き込み直前まで有効だとみなした。

停止指示の後でも、実行直前に権限を再確認しなければ古い要求が通る。

02 判断のズレ

起動時に許可済み ≠ **実行時も許可中**

確認すること： 外部変更の直前に現在の権限を確認したか

03 直したこと

外部変更の直前に停止状態・担当・期限・世代を再照合。停止前に発行された古い要求は、実行経路で拒否する。

04 確認結果

設計済み｜全経路は未検証

実行直前に権限を再確認し、古い要求を拒否する設計を追加。全経路での強制は未検証。

止めすぎて、復旧に必要な修理まで止まった。

統括 AI | 制御システム | 実装 AI

出典：内部 I1 §10.2 / GitHub G1

01 起きたこと

安全のため全変更を止め、承認済みの復旧作業まで止めた。

通常変更は停止。承認済み修理と非競合作業だけ通す。

02 判断のズレ

全部止めれば安全 ≠ **復旧も進められる**

確認すること： 停止対象と、通してよい修理を分けたか

03 直したこと

停止対象・解除条件・責任者を限定。通常変更は拒否し、承認済み修理と非競合作業だけ通す。

04 確認結果

統合を確認｜復旧時間は未測定

関連コードの統合と CI 成功は確認済み。復旧時間が短くなったかは未測定。

認証で止まっていたのに、コードを直し始めた。

実装 AI　統括 AI

出典：内部 I1 §11,14

01 起きたこと

認証・利用枠の起動前拒否を、コード不具合として扱った。

失敗段階を「起動前拒否 / 実行中不具合 / 結果不明」に分ける。

02 判断のズレ

エラーが出た ≠ **コードに不具合がある**

確認すること： どの段階で失敗したか

03 直したこと

ログを「起動前拒否・実行中不具合・結果不明」に分類。原因に応じて、利用枠・認可・実装修正へ振り分ける。

04 確認結果

設計済み｜誤修正件数は未測定

失敗段階を分類して原因別に振り分ける規則を設計。不要修正の減少は未測定。

同じ Mac でも、使った Node.js が違った。

実装 AI

出典：会話 C1 / 内部 I1 §11

01 起きたこと

同じ Mac なら、同じ実行環境だとみなした。

Node 16.13.1 では拒否 / Node 24.11.1 では起動を確認。

02 判断のズレ

端末が同じ ≠ **実行ファイルと版も同じ**

確認すること： 実行ファイルの場所とバージョンを実測したか

03 直したこと

起動前に実行ファイル、版、作業場所、必要設定を確認。手動成功を別経路の成功へ一般化しない。

04 確認結果

起動確認済み｜他経路は未確認

Node 24.11.1 へ切替後、コマンド起動を確認。他の実行経路は未確認。

「 ready 」と表示された。それだけで、作業できると判断した。

統括 AI　実装 AI

出典：内部 I1 §11,14

01 起きたこと

ready 表示だけで、接続・認証・権限・実行まで可能だと判断した。

ready ≠ 接続 ≠ 認証 ≠ 認可 ≠ 実行 ≠ 結果確認

02 判断のズレ

ready と表示された ≠ 必要操作まで実行できる

確認すること： 接続・認証・認可・実操作・結果を個別に確認したか

03 直したこと

ready 、接続、認証、認可、実行、結果を別の証拠として記録。必要な段階まで実操作で確かめる。

04 確認結果

設計済み｜接続全体は未監査

ready から結果確認までを別証拠に分ける規則を設計。現行の全接続は未監査。

禁止ルールを書いた。でも、別経路から実操作できた。

制御システム　統括 AI　実装 AI

出典：内部 I1 §6,10,14 / GitHub G2

01 起きたこと

禁止ルールはあったが、実操作が別経路から抜けた。

ルールの存在ではなく、すべての外部変更経路が制御点を通ることが必要。

02 判断のズレ

禁止ルールが存在する ≠ **禁止操作を強制的に止められる**

確認すること： すべての外部変更経路が制御点を通るか

03 直したこと

制御条件を実操作の境界へ接続。禁止操作の拒否と正常操作の許可を両方試し、未保証の経路を明記する。

04 確認結果

コード上は確認｜実稼働は未確認

制御条件の実装はコード上で確認。実稼働で全経路を止められるかは未確認。

05 レビュー・証拠

対象版・必須検査・必要証拠を先に固定する。変更の影響範囲だけ再確認する。

判断基準 **対象版・必須検査・必要証拠を先に固定する。変更の影響範囲だけ再確認する。**

典型的な失敗の流れ

01 緑表示・ラベルをそのまま信じる
›
02 対象版・必須検査を確かめない
›
03 未確認の成果を承認する

該当ケース CASE 17 CASE 18 CASE 19 CASE 20 CASE 22

5 件の事例を、レビュー・証拠としてまとめて読む。

1 か所直すたび、確認済みの証拠まで白紙に戻した。

レビュー AI　実装 AI

出典：内部 I1 §2,7.2,9 / GitHub G1

01 起きたこと

一部修正のたびに、影響しない過去証拠まで無効にした。

A ・ C は保持し、修正した B と影響範囲だけ再確認する。

02 判断のズレ

版が変わった ≠ **すべての証拠が無効になった**

確認すること： 変更の影響を受ける証拠はどれか

03 直したこと

旧証拠を保持し、変更差分・元の指摘・依存先だけ再確認。最新候補の必須 CI は毎回実行する。

04 確認結果

統合を確認｜時間短縮は未測定

関連 PR の統合と CI 成功は確認済み。再レビュー時間の削減効果は未測定。

同じ PR 番号なので、変更後の版にも古い承認を流用した。

統括 AI　レビュー AI　実装 AI

出典：内部 I1 §9,14 / 一次資料 E2

01 起きたこと

PR 番号が同じなので、変更前の承認を変更後にも流用した。

PR #42 / SHA A は承認済みでも、追加変更後の SHA B は別判定が必要。

02 判断のズレ

同じ PR 番号 ≠ **承認された内容も同じ**

確認すること： 承認 SHA と公開候補 SHA が一致するか

03 直したこと

承認を SHA へ固定し、追加変更には新しい判定を要求。ソース → 成果物 → 配備先 の対応も記録する。

04 確認結果

設計済み｜未確認版防止は未測定

承認を SHA へ固定する設計は済み。未確認版の公開を防げた件数は未測定。

High / Low のラベルだけで、停止と続行を決めた。

レビュー AI　統括 AI

出典：内部 I1 §2,8

01 起きたこと

High / Low ラベルだけで、停止・続行を決めた。

重大度・確信度・受入可否は別の軸。

02 判断のズレ

重大度が高い / 低い ≠ **今回止める / 進めるべき**

確認すること： 影響・根拠・受入条件への影響を確認したか

03 直したこと

失敗経路・影響・根拠・変更との関係を示す。重大度、確信度、受入可否を別々に判定する。

04 確認結果

設計済み｜判断精度は未測定

重大度・確信度・受入可否を分ける規則を仕様化。誤停止や見逃しの減少は未測定。

CI の表示は緑。でも、必須テストは走っていなかった。

統括 AI　レビュー AI　実装 AI

出典：内部 I1 §9 / 一次資料 E2

01 起きたこと

CI の緑表示だけで、必須テストも実行済みと判断した。

ビルド PASS / 認証テスト SKIPPED / 文書検査 PASS

02 判断のズレ

CI が緑 ≠ **必要な検査がすべて実行済み**

確認すること： 必須検査項目の実行・スキップ状態を照合したか

03 直したこと

対象版の必須試験一覧と検査ジョブの実行記録を突合。スキップ理由を確認し、未実施を PASS として扱わない。

04 確認結果

仕様確認済み｜運用効果は未測定

CI のスキップ表示仕様は確認済み。検査漏れの減少は未測定。

CI が通った。それだけで、独立レビューも終わった扱いにした。

統括 AI　レビュー AI　実装 AI

出典：内部 I1 §3,5,14

01 起きたこと

CI 成功だけで、必須だった独立レビューも完了したと判断した。

自動検査と、別担当の独立判定は別の完了条件。

02 判断のズレ

自動検査が成功 ≠ **独立レビューも完了**

確認すること： 誰が、どの版を、何を根拠に判定したか

03 直したこと

元の依頼・対象 SHA ・差分・実行済み検査を別担当のレビュー AI へ渡す。自動検査と独立判定を別記録にする。

04 確認結果

設計済み｜独立性の効果は未測定

実装・自動検査・独立レビューを分離する設計は済み。精度向上は未測定。

06 収束・優先順位

顧客復旧を先に終わらせる。改善の予算と受入条件は目的ごとに固定する。

判断基準 **顧客復旧を先に終わらせる。改善の予算と受入条件は目的ごとに固定する。**

典型的な失敗の流れ

01 改善を足し続ける › 02 受入条件・試行予算が膨らむ › 03 目の前の顧客復旧が遅れる

該当ケース CASE 15 / CASE 16 / CASE 24

3 件の事例を、収束・優先順位としてまとめて読む。

PR や担当を替えながら、同じ修正を繰り返した。

統括 AI　実装 AI　レビュー AI

出典：内部 I1 §7,13

01 起きたこと

PR や担当が変わるたび、同じ目的の試行回数をリセットした。

PR / 担当 / 会話が変わっても、目的 ID と累積予算は 1 つ。

02 判断のズレ

PR / 担当 / 会話が変わった ≠ **目的の試行予算もリセットできる**

確認すること： 同じ目的 ID に試行履歴を累積しているか

03 直したこと

予算・指摘・未解決時間を目的 ID へ累積。手法、新証拠、承認済み追加枠が変わるときだけ再計画する。

04 確認結果

設計済み｜収束時間は未測定

目的単位で試行予算を累積するルールは設計済み。収束時間の改善は未測定。

レビューのたびに、合格条件が増えた。

レビュー AI　統括 AI

出典：内部 I1 §4,7–8 / GitHub G1 / 一次資料 E1

01 起きたこと

レビューの途中で、今回の合格条件を追加し続けた。

着手時の受入条件 A/B/C に、 D/E/F/G を足し続けた。

02 判断のズレ

新しい欠点を見つけた ≠ **今回の合格条件に追加すべき**

確認すること： 着手時の受入条件と非対象を守っているか

03 直したこと

用途・非対象・必須検証を着手時に固定。重大な新事実以外の改善は別タスクへ分ける。

04 確認結果

一部実装済み｜収束効果は未測定

受入条件を途中で増やさない考え方は一部実装済み。レビュー収束の効果は未測定。

再発防止を増やすほど、目の前の不具合修正が遠のいた。

統括 AI | 制御システム | 実装 AI

出典：内部 I1 §3,12,15–16 / 一次資料 E4

01 起きたこと

再発防止の基盤改善が膨らみ、目の前の顧客修復を遅らせた。

今の修理は最小変更で進め、広い再設計は別目的へ切り出す。

02 判断のズレ

仕組みを増やす ≠ 顧客復旧も速くなる

確認すること： 今の修理に本当に必要な変更か

03 直したこと

顧客復旧に必要な最小変更を先行。広い基盤改善は別目的・別の完了条件へ切り出す。

04 確認結果

設計済み｜復旧時間は未測定

顧客復旧と基盤改善を分ける設計は済み。復旧時間の改善は未測定。

24 事例を、 8 つの運用原則に圧縮する。

個別対応で終わらせず、次から守るルールにする。

24

24 の事例

↓

8

8 つの原則

1 ✓ 完了条件を先に決める

2 ▤ 状態と再開点を残す

3 ◎ 役割を分ける

4 🔒 権限を制御する

5 ◌ 独立レビューを入れる

6 ▤ 証拠を残す

7 ▥ 優先順位で収束させる

8 ◫ ルールを文書化する

⚑ 原則は短く、**現場で使える形**にする。

毎回の仕事は、この 6 問で確認する。

上から順に確認し、 CURRENT_BLOCKER だけ対象 scope を止める。

1 完了・受入
誰が何をできれば完了か?

2 状態・再開
今の版・担当・停止状態はどうなっているか?

3 担当・委任
誰が判断し、どこへ書くか?

4 実行環境・制御
実行直前の環境・権限・経路は正しいか?

5 レビュー・証拠
対象版と必要証拠は一致しているか?

6 収束・優先順位
顧客目的へ近づいているか? 改善を増やしすぎていないか?

3 つの例外

結果不明
再送しない。操作 ID で外部記録を照合する。

停止中
再開条件と未完了作業を残す。

依頼者判断
新規費用・契約・不可逆変更だけ依頼者が判断する。

再開記録: 目的 / 対象版 / 担当 / 残件 / 停止 / 結果不明 / 予算 / 再開条件

次は、この 6 つの数字で効果を測る。

基準値を取る → 4 週間運用する → 前後を比べる。動かない仕組みは簡素化する。

完了・受入

誤完了率

完了報告のあと、利用者確認で未達だった割合

状態・再開

再開ロス

引き継ぎ後に、状態を調べ直した時間

担当・委任

依頼者割込み

委任範囲内なのに依頼者へ戻した回数

実行環境・制御

制御外変更

制御点を通らず外部変更した件数

レビュー・証拠

再レビュー回数

1 つの目的が承認までに再レビューされた回数

収束・優先順位

復旧時間

利用者影響の発生から受入完了までの時間

測定の順番：基準値 → 4 週間運用 → 前後比較 → 削除・簡素化・再設計

重大度と、今回を止める判断を分ける。

Severity は影響の大きさ。 CURRENT_BLOCKER は固定 DoD を満たせるかで決める。

	固定 DoD への影響あり	固定 DoD への影響なし
HIGH / P0 ・ P1	**CURRENT_BLOCKER** 対象 scope だけ止め、最小修正する。	**FOLLOW_UP** 重大度・回避証拠・再判定条件を残す。
LOW / MEDIUM	**CURRENT_BLOCKER** 軽微でも受入未達なら、今回直す。	**FOLLOW_UP / IGNORE** 関連があれば後続へ。無関係なら今回から外す。

⚑ **停止判断： Severity × 根拠の確かさ × 固定 DoD への影響。ラベル単独では止めない。**

Finding は、 3 つの状態へ即時分類する。

レビューで問題を見つけた瞬間に、状態と次の動作を固定する。

01 CURRENT_BLOCKER

今回の固定 DoD を満たせない

- 必須受入条件が未達
- 必須安全条件が未達
- 現在の事故経路が成立

対象 scope だけ停止 → 最小修正 → 再受入

02 FOLLOW_UP

重要でも今回を閉じられる

- 回避で目的と安全性を維持
- 将来 hardening が中心
- 現在 DoD には無関係

重大度 / 担当 / 回避証拠 / 再判定 / 再昇格を記録

03 IGNORE

今回の目的に関係しない

- 対象外の既存問題
- 別目的の改善案
- 証拠のない一般論

現在の完了判定から外す

分類は必須入力にする。新しい証拠が DoD を崩した時だけ、 FOLLOW_UP を CURRENT_BLOCKER へ昇格する。

A は完了を閉じる。 B は次の問題を減らす。

同じ意味領域は single-writer 。独立調査と準備は並行して進める。

A / CONVERGENCE

現在の固定 DoD を閉じる

固定候補 → 受入 QC → CURRENT_BLOCKER だけ最小修正 → COMPLETE

B / PREPARATION

後から出る問題を先回りする

FOLLOW_UP 調査 → 回避策 → 限定修正準備 → 検証証拠を揃える

B → A 割込み条件

- 現在 DoD 未達
- 必須安全条件未達
- 現在の事故経路が成立

B の是正設計を A の新しい前提へ追加しない。 A の完了条件を守り、 B は準備に徹する。

設計したルールを、停止・続行判断へ強制する。

文書化で終えず、 Finding から次の動作までを毎回同じ手順で通す。

1 Finding

問題と証拠を受け取る

2 Classify

三択を必須で付ける

3 Act

分類に対応する動作だけ行う

4 Record

Evidence と再判定条件を残す

判断精度と throughput を一緒に測る

完了時間

DoD 固定 → COMPLETE

再レビュー回数

1 目的あたり

Reopen 率

完了後に再オープン

B→A 昇格率

FOLLOW_UP が blocker 化

⚑ **Reopen 率や B→A 昇格率が上がるなら、 FOLLOW_UP 判定が甘い。速度と品質を同じ指標群で見る。**

この資料で確認できた範囲を、先に区切る。

記録された事実・分析・運用効果を、同じ確度で扱わない。

C この会話

依頼者の指示・評価 / 制作過程 / 2026-09-23 収束判断の記録

I 内部資料

統合仕様 v3 / A ・ B 運用ルール / スライド制作契約

G 当時の GitHub

PR #125 の統合 / PR #100 の当時状態

E 一次資料

Google / GitHub / AWS / Anthropic / 一般原則の補助参照

この資料で言えること

- 記録された出来事
- 依頼と行動の差から読める判断のズレ
- 当時確認した統合・設定の状態

この資料の対象外

- AI の内面や意図の断定
- 現在も同じ不具合があるという判断
- 未測定の運用効果を「成功」とすること

現行システムの監査は対象外。現在の状態は、必要なときに改めて確認する。

失敗を、次の運用ルールに変える。

24 事例から見えたことを、明日から使える原則と行動に落とし込む。

◎ 見えたこと

- ✓ 失敗は設計と運用判断の両方で起きる
- ✓ 役割が曖昧だと判断が混線する
- ✓ 状態と証拠がないと再開できない

8 8 つの原則

- ✓ 完了条件を先に決める
- ✓ 状態と再開点を残す
- ✓ 役割を分けて、権限を制御する
- ✓ 独立レビューと優先順位で収束させる

→ 次にやること

- ✓ 自分の現場の失敗を 6 領域に並べる
- ✓ 足りない原則を洗い出す
- ✓ 三択と A/B レーンを運用判断へ強制する

⚑ **反省で終わらせず、再発防止の仕組みに変える。**