Claude Code × 社内ナレッジ
RAG構築ガイド



社内の設計書・コーディング規約・障害履歴を活用し、開発品質と生産性を高める

Start Challenge Consulting co., ltd.

2026年8月　発行

Confidential　本資料の無断複製・二次配布を禁じます

開発部門・情報システム部門
責任者向け　ホワイトペーパー

Start Challenge
Consulting co., ltd.
挑む力が、世界を照らす

はじめに｜なぜ「一般的な回答」に留まるのか

Claude Codeは汎用的なコード生成には強い一方、社内固有の設計思想・規約・障害履歴を参照できない。RAGによる社内ナレッジ接続が、回答を「一般解」から「自社の正解」へ変える前提条件となる。

現状｜Claude Code導入後に起きていること

01　回答が一般論に寄る

公開情報から学習した「よくある実装」は提示できるが、自社の共通ライブラリ・命名規則・アーキテクチャ判断は前提に含まれない。

02　社内固有のルールと過去の知見が届かない

設計思想、レビュー観点、過去の障害から生まれた禁止事項は、いずれも社内文書の中にのみ存在する。

03　補完は人手で行われている

担当者が都度、設計書や規約をプロンプトへ貼り付けており、その手間と精度が個人差として表面化している。

目的｜Claude CodeとRAGを組み合わせる

①回答の前提を社内情報へ固定する

依頼のたびに関連する社内文書を検索し、根拠としてClaude Codeへ供給する。

②個人差を仕組みで吸収する

「誰が使っても同じ情報源を参照する」状態をつくり、チーム間の品質差を縮小する。

③権限と監査を保ったまま活用する

利用者の閲覧権限を引き継いだ検索により、機密情報の統制を維持する。

④更新を回答へ即時反映する

モデルを再学習させず、文書の更新だけで最新仕様に追随させる。

本資料で分かること

① Claude Code単体の限界と、RAGが解決する範囲

② 連携アーキテクチャとデータ整備の実務手順

③ セキュリティ・アクセス制御の設計論点

④ 品質評価の方法と段階的な導入ロードマップ

事実：Claude Codeは社内文書を既定では参照しない／解釈：品質差はプロンプト依存に起因／提案：RAGによる社内ナレッジ接続（提案）

Claude駸odeとは

Claude Codeはターミナル上でリポジトリ全体を横断し、調査から生成・修正・テストまでを自律的に実行するエージェントであり、行単位で候補を出す補完ツールとは作業単位そのものが異なる。

実行環境

ターミナル（CLI）上で利用するAIコーディングエージェント。開発者が普段使う作業環境の中で対話しながら作業を進める。

支援範囲

コードの生成・修正に加え、既存実装の調査、テストの作成と実行までを一連の作業として支援する。

横断的な作業

単一ファイルではなくリポジトリ全体を対象に、関連箇所を探索しながら影響範囲を踏まえた変更を行える。

自律的なタスク実行

目的を伝えると、必要な調査・変更・検証の手順を自ら組み立てて実行し、結果を報告する。

単なるコード補完ツールとの違い

観点	一般的なコード補完ツール	Claude駸ode（エージェント型）
作業の単位	行・関数単位で次の記述候補を提示する	タスク単位（調査 → 実装 → 検証）で完結させる
参照する範囲	編集中のファイル・近傍のコードが中心	リポジトリ全体を横断して関連箇所を探索する
実行の形態	エディタ上での入力補助にとどまる	ターミナル上でコマンド実行・テスト実行を伴う
人の関与	提示された候補を都度、採用・修正する	目的と制約を指示し、成果物をレビューする

作業単位が大きいほど、前提として与えるべき社内情報の量と精度が成果を左右する。

事実：構成案に記載のClaude Code概要／解釈：補完ツールとの差分は「作業単位」と「参照範囲」に集約されるとの整理

Claude Code単体利用の限界

単体利用では社内固有の前提が欠落するため、回答品質が担当者のプロンプト作成力に依存し、チーム間のばらつきと手戻りという形で開発生産性を押し下げる。

01 設計思想の断絶

社内の設計思想や業務ルールを把握できず、技術的には正しくても自社の運用に合わない実装が提案される。

結果：レビュー段階での指摘・作り直しが発生する。

02 情報の鮮度

最新の社内ドキュメントが回答に反映されず、改訂済みの規約や廃止済み仕様が考慮されない。

結果：古い前提に基づく実装が混入する。

03 過去知見の断絶

過去の障害や類似案件を参照できず、一度発生した不具合パターンを回避する判断が働かない。

結果：同種の障害・指摘が再発する。

04 品質のばらつき

与える情報の質がチーム・個人ごとに異なるため、同じ依頼でも回答品質が揃わない。

結果：成果が属人化し、横展開できない。

05 運用負荷

必要な情報を毎回プロンプトへ貼り付ける手間が発生し、文書を探す時間が削減効果を相殺する。

結果：利用が定着せず形骸化する。

業務・経営影響

これら5つの限界は個別の不便ではなく、**「レビュー工数の増加」「手戻りの発生」「AI投資が生産性へ結びつかない」**という形で集約される。

解決の焦点は、モデルの性能ではなく**「AIへ与える社内情報の供給経路」**の設計にある。

事実：構成案に記載の5つの限界／解釈：影響は工数増・手戻り・定着不全に集約されるとの整理（解釈）

開発現場に散在する社内ナレッジ

開発ナレッジはコード・設計書・チャット・障害報告に分散しており、「どれが正式な情報源か」が定義されていないこと自体が、AI活用以前のボトルネックとなっている。

01　ソースコード（GitHub／GitLab）

実装の一次情報。命名規則・実装パターン・修正履歴が蓄積される一方、意図や背景はコードに残らない。

更新頻度：高／正確性：高／機密性：高

02　コーディング規約・ガイドライン

品質判断の基準。改訂版と旧版が併存しやすく、どれが現行かの判別が難しい。

更新頻度：低／正確性：高／機密性：中

03　要件定義書・基本／詳細設計書

システム仕様の根拠。案件単位でフォルダが分かれ、版管理と失効判定が課題となる。

更新頻度：中／正確性：中／機密性：高

04　Confluence・Notion・SharePoint

技術文書・運用手順の中心。正式文書と個人メモが同じ場所に混在しやすい。

更新頻度：中／正確性：中／機密性：中

05　Slack・Teamsの質疑応答

暗黙知が最も蓄積される場。一方で結論と検討途中の発言が区別されず、誤参照リスクが高い。

更新頻度：高／正確性：低／機密性：中

06　障害報告書・問い合わせ・議事録

再発防止に直結する知見。定型フォーマットで残る一方、検索されずに埋没しやすい。

更新頻度：中／正確性：高／機密性：高

論点　ナレッジは「不足」しているのではなく「散在」している。RAG構築の実質的な作業は、収集そのものよりも**正式な情報源の定義・鮮度管理・機密区分の付与**にある。

事実：構成案に記載の6分類／解釈：各分類の更新頻度・正確性・機密性は一般的傾向の整理（解釈・要現場検証）

RAG（検索拡張生成）の基本的な仕組み

RAGは依頼に関連する社内文書を検索し、その内容をコンテキストとしてClaude Codeへ渡す仕組みであり、モデルを再学習させずに最新の社内知識を回答へ反映できる。

STEP 1　依頼

開発者がClaude Codeへ実装・調査・レビューを依頼する。依頼文そのものが検索のクエリとなる。

▶

STEP 2　検索

依頼に関連する社内情報を、キーワードと意味の両面から検索し、上位の候補を抽出する。

▶

STEP 3　文脈付与

検索結果を出典情報とともにClaude Codeへコンテキストとして提供する。

▶

STEP 4　生成

社内ナレッジを踏まえた回答・コードを、参照元を明示したうえで生成する。

※ モデルの重みは変更されない。反映されるのは「その依頼のときに渡された情報」であり、文書を更新すれば次回の回答から反映される。

ファインチューニングとの違い

観点	RAG（検索拡張生成）	ファインチューニング
知識の更新	検索対象の文書を更新すれば、次の依頼から反映される	反映には再学習が必要で、更新の都度コストが発生する
出典の提示	参照した文書名・更新日を回答に添えられる	どの学習データに基づく出力かを特定しにくい
権限制御	検索時に利用者の閲覧権限を適用できる	学習済みモデルに権限の概念を持たせにくい
主な用途	最新かつ機密性のある社内情報の参照	出力の文体・形式など振る舞いの調整

事実：構成案に記載のRAG概要／解釈：RAGとファインチューニングの差分整理（設計上の性質を一般論として整理）

Claude Code × RAGで実現できること

効果が出やすいのは「社内情報を探す時間が長く、正解が社内文書に明記されている業務」であり、コード生成そのものよりも調査・レビュー・引き継ぎ領域で効果が先行する。

01　社内標準に沿ったコード生成

コーディング規約と共通ライブラリを参照させ、命名・例外処理・ログ出力を社内標準へ揃えた実装を生成する。

主な参照先：コーディング規約／共通部品

02　類似実装を参照した改修

過去の同種案件の実装と設計判断を提示し、ゼロから設計し直すことなく整合性のある改修方針を導く。

主な参照先：ソースコード／設計書

03　障害履歴を考慮したレビュー

過去に発生した不具合パターンと再発防止策を突き合わせ、レビュー観点として明示的に指摘させる。

主な参照先：障害報告書／改修履歴

04　仕様に基づくテストコード生成

実装からではなくシステム仕様から観点を起こし、境界値・異常系を含むテストケースを作成する。

主な参照先：詳細設計書／API仕様書

05　レガシーシステムの仕様調査

有識者が不在の領域について、コードと設計文書を突き合わせ、現行仕様と改修時の影響範囲を整理する。

主な参照先：ソースコード／旧設計書

06　新任メンバーへのナレッジ提供

「どこに何が書かれているか」を出典付きで案内し、有識者への質問に依存した立ち上がりを解消する。

主な参照先：開発ガイドライン／技術文書

初期スコープの考え方

6つを同時に狙わない。**「参照すべき文書が明確」かつ「現在の調査時間が長い」**ユースケース（例：01・03・05）から着手し、効果測定が可能な単位で範囲を広げる。ユースケースの選定が、後続のデータ選定（P9）を規定する。

事実：構成案に記載の6ユースケース／提案：着手順は参照先の明確さと調査時間の長さで判断（提案）

全体アーキテクチャ

Claude Codeと社内データソースの間に「検索インターフェース」「認証・認可」「検索基盤」の3層を置き、権限を保ったまま関連情報のみを返す構成とする。



構成要素と主な役割

Claude Code
依頼を解釈し、検索実行の要否を判断して結果を利用する。

MCPサーバー／社内検索API
社内検索を「ツール」としてClaude Codeへ提供する接続口。

認証・認可基盤
利用者の権限を引き継ぎ、閲覧可能な範囲のみを検索対象とする。

検索エンジン／ベクトルDB
キーワード一致と意味的類似の双方から候補を抽出する。

埋め込みモデル
文書とクエリをベクトル化する。日本語・コードの精度が選定要件。

ログ・監査・フィードバック基盤
検索・参照・生成の記録を残し、品質改善と統制の両面に用いる。

設計上の要点：Claude Codeにデータソースを直接接続せず、必ず認証・認可層を経由させることで、権限逸脱と情報漏えいの経路を一本化する。

連携対象データの選定

対象データは「網羅」ではなく利用目的からの逆算で決める。正確性が高く更新管理された文書から始めることが、初期の回答精度と信頼獲得に直結する。

選定の4原則

原則1　利用目的から必要なデータを逆算する
先に対象ユースケース（P7）を確定し、その回答に必要な文書だけを対象とする。データ起点で始めない。

原則2　正式文書と参考情報を区別する
規約・設計書などの正式文書と、チャットや個人メモを同列に扱わない。区分をメタデータとして保持する。

原則3　更新頻度・正確性・機密性を評価する
3軸で棚卸しし、正確性が低い、または機密性が高く統制が未整備の情報は初期対象から除外する。

原則4　最初からすべてのデータを取り込まない
対象を広げるほど検索ノイズと誤回答が増える。範囲拡大は評価（P15）で精度を確認した後に行う。

MVPで優先すべきナレッジ

① コーディング規約
判断基準が明文化され、正解が一意。効果が最も検証しやすい。

② システム設計書
仕様調査・影響範囲確認の起点。版管理が前提条件となる。

③ API仕様書
構造化されており分割しやすく、テスト生成との相性が高い。

④ 障害・改修履歴
再発防止という明確な価値があり、効果を説明しやすい。

初期対象から外すことを推奨する情報　チャットの雑談・検討途中の議事メモ・失効した旧仕様・個人フォルダの資料。
これらは正確性の担保が難しく、誤回答の主要因となるため、正式化・区分整理を経てから追加する。

事実：構成案に記載のMVP優先ナレッジ4種／提案：選定4原則および初期除外対象の整理（提案）

データ収集と前処理

前処理の目的は形式変換ではなく「検索対象として信頼できる状態」への正規化であり、機密判定と失効文書の識別を通過しない文書は取り込まない運用とする。

STEP 1　文書の取得

各データソースから対象文書を取得する。API連携による定期同期を基本とし、手動アップロードは例外運用とする。

論点：同期頻度・差分取得の可否・削除の反映

STEP 2　形式の統一

PDF・Word・Markdown・HTMLなどをテキストへ統一する。表・図中の文字は欠落しやすく、抽出精度の確認が必要。

論点：スキャンPDF・図表内テキストの扱い

STEP 3　ノイズ除去

ヘッダー・フッター・目次・定型文・重複文書を除去する。全ページ共通の文言は検索ノイズの主要因となる。

論点：ほぼ同一の版が複数存在する場合の統合

STEP 4　機密情報の検出

個人情報・認証情報・顧客固有情報を検出し、除外またはマスキングする。**取り込み前の必須関門**として位置付ける。

論点：検出漏れ時の是正手順と責任分担

STEP 5　属性の付与

タイトル・作成者・更新日・所管部署を付与する。回答時の出典表示とフィルター条件の基礎となる。

論点：属性が欠落した文書の取り扱い方針

STEP 6　失効文書の識別

古い文書や廃止済み仕様を識別し、除外または「参考」区分へ降格する。**誤回答の最大要因**となる領域。

論点：現行版を判定する責任者の明確化

運用上の要点　前処理は一度きりの作業ではなく継続運用である。文書の追加・改訂・廃止に追随する更新プロセスと、その責任者（文書オーナー）を定義しない限り、時間の経過とともに回答精度は必ず劣化する。

事実：構成案に記載の前処理6項目／提案：STEP4・6を必須関門とする運用（提案）

チャンク分割とメタデータ設計

文字数のみで機械的に分割すると仕様が文の途中で切断され、断片的な情報から誤った実装が導かれる。見出し・機能・クラス単位という意味の区切りで分割する。

分割方式の比較

方式A　固定文字数のみで分割（非推奨）

・「エラー時は」で切れ、後続の処理内容が別チャンクへ分離する
・表の見出し行とデータ行が分断され、値の意味が失われる
・どのシステム・どの機能の記述かが、チャンク単体からは判別できない

帰結：検索は成功しても、渡される文脈が不完全なため誤った実装が生成される

方式B　意味の区切りで分割（推奨）

・**文書**：見出し階層（章・節・機能単位）で区切り、親見出しを各チャンクへ引き継ぐ
・**ソースコード**：クラス・メソッド単位で区切り、ファイルパスと定義名を保持する
・**API仕様書**：エンドポイント単位で区切り、リクエスト・レスポンスを同一チャンクに収める

帰結：チャンク単体で意味が完結し、出典提示とレビューが可能になる

推奨メタデータ設計

項目	用途
システム名	対象システムへ検索範囲を限定する
プロジェクト名	案件単位での絞り込みに用いる
文書種別	規約・設計書・障害報告等を区別する
バージョン	現行版と旧版を判別する
更新日	鮮度の重み付けと出典表示に用いる
機密区分	外部送信可否・参照可否を判定する
閲覧可能な部署・役職	検索時のアクセス制御に用いる（P14）
正式／参考区分	根拠として提示可能かを区別する

設計の要点　メタデータは検索精度（P12）とアクセス制御（P14）の双方を支える共通基盤である。
後から遡って付与する作業は負荷が大きいため、取り込み時点で必須項目として設計しておくこと。

事実：構成案に記載の推奨メタデータ例／提案：意味単位分割の採用と「正式／参考区分」の追加（提案）

検索精度を高める設計

ベクトル検索単独では、エラーコードや関数名といった固有語の一致に弱い。キーワード検索との併用・メタデータ絞り込み・リランキングの三段構えで精度を確保する。

検索パイプライン

①キーワード検索
関数名・エラーコード・固有名詞の完全一致に強い

②ベクトル検索
表現が異なる同義の記述・意図の近い文書を拾える

▼

③ハイブリッド統合・メタデータフィルター
両者の結果を統合し、システム名・文書種別・現行版・閲覧権限で対象を限定する

▼

④リランキング
依頼内容との関連度で並べ替え、上位の少数のみをClaude Codeへ渡す（渡す量ではなく順位が精度を決める）

▼

出典・更新日を添えてClaude Codeへ返却

設計上の判断ポイント

検索対象で手法を使い分ける
ソースコードは構造・識別子の一致、文書は意味的類似が有効。同一の検索設定を全体へ適用しない。

鮮度を順位へ反映する
現行版を優先し、失効文書は既定で除外する。旧版参照は明示指定時のみ許可する。

回答へ参照元・更新日を表示する
利用者が根拠を検証できることが、AI回答を業務で採用するための前提条件となる。

渡しすぎない
関連度の低い文書を大量に渡すと、かえって回答が曖昧になる。上位数件に絞る。

検索精度の改善で最初に確認すべき3点

1. そもそも対象文書が取り込まれているか
2. 検索はできているが順位が低いのか
3. チャンクが分断され意味を失っていないか

事実：構成案に記載の検索設計6項目／提案：4段パイプラインとしての構成化、および改善時の切り分け手順（提案）

Claude Codeとの接続方法

接続は「社内検索をツールとして提供する」ことに加え、いつ検索するか・根拠をどう示すか・見つからない場合どう振る舞うかという利用ルールの定義まで含めて初めて機能する。

接続方式

方式1　MCPで社内検索機能をツールとして提供する

社内検索を標準化された「ツール」としてClaude Codeへ登録する。Claude Code側が必要と判断した時点で自律的に検索を呼び出せるため、開発者が手順を意識せずに利用できる。

方式2　社内APIを経由してナレッジを取得する

既存の社内検索APIを呼び出して結果を取得する。既存資産を活用でき統制も掛けやすいが、呼び出しの手順・条件を利用側で定義する必要がある。

方式3　プロジェクト設定ファイルで利用ルールを定義する

リポジトリ単位の設定ファイルへ、参照すべき情報源・検索の実行条件・出典提示の作法を記述する。方式1・2と併用する前提の「運用ルールの実装」に相当する。

接続方式の選定は既存環境に依存するが、**「情報が無い場合は推測させない」ルールの実装は方式を問わず必須**である。
このルールが欠落すると、それらしい仕様を生成する誤回答が発生し、RAG導入の信頼性そのものが損なわれる。

定義すべき利用ルール

項目	定義内容
検索の実行条件	仕様・規約・障害に関わる依頼では必ず検索を実行する。逆に一般的な文法・言語仕様の質問では検索しない。
参照すべき情報源	正式文書を優先し、参考情報は補足としてのみ扱う。対象システム・現行版に限定する。
根拠の提示	回答には参照した文書名・該当箇所・更新日を必ず併記する。
情報が無い場合の挙動	該当情報が見つからない場合は推測で補わず、「社内文書に記載が見当たらない」旨を明示させる。
矛盾時の扱い	複数文書で記述が異なる場合は、勝手に一方を採用せず差異を提示して判断を人へ委ねる。

事実：構成案に記載の接続手段（MCP／社内API／設定ファイル）／提案：利用ルール5項目の定義（提案・要環境検証）

セキュリティとアクセス制御

RAGは「本人が閲覧できない情報をAI経由で閲覧できてしまう」構造的リスクを内包する。検索時に利用者権限を引き継ぐ設計が、導入可否を左右する必須要件となる。

① アクセス制御

利用者権限を引き継いだ検索
システム共通の管理者権限で検索させない。実行者本人の権限で検索範囲を決定する。

部署・案件・役職単位の制御
メタデータの「閲覧可能な部署・役職」（P11）を検索条件へ適用し、対象外を結果から除外する。

② 情報の送信範囲

プロンプトへ渡す範囲の制限
機密区分に応じ、検索結果を渡すか、所在の提示のみに留めるかを分岐させる。

認証情報のマスキング
APIキー・接続情報・パスワードは検出のうえ除去する。

外部送信不可データの整理
顧客データ・個人情報など、外部モデルへ送信できない区分を事前に定義する。

③ 監査と権限運用

監査ログの取得
誰が・いつ・何を検索し・どの文書を参照して・何が生成されたかを記録する。

権限の更新運用
退職者・異動者の権限を速やかに反映する。人事情報との連携が実効性を担保する。

想定リスクと対応方針

想定リスク	発生経路	対応方針
権限逸脱による情報閲覧	共通権限での検索により、本人が閲覧不可の文書が回答に含まれる	利用者権限を引き継いだ検索を必須とする
機密情報の外部送信	検索結果に認証情報・個人情報が含まれたまま渡される	取り込み時の検出（P10）＋送信前マスキングの二重化
利用実態が追跡できない	監査ログが未整備で、参照経路と責任を事後に特定できない	検索・参照・生成の3点をログ化し保管期間を定める

これらは技術要件であると同時に、情報セキュリティ部門・人事部門との合意事項である。PoC段階から関係部門を巻き込むことを推奨する。

事実：構成案に記載のセキュリティ7項目／解釈：3階層（制御・送信範囲・監査）への整理／提案：リスク対応方針（提案・要自社規程との突合）

回答品質を評価する方法

品質を「良い／悪い」で語らず、検索・根拠・準拠の3層に分解して測る。どの層で失敗したかを特定できる評価設計が、改善の速度をそのまま決定する。

評価の3層と確認項目

第1層　検索	・正しい文書を検索できたか（想定していた文書が上位に含まれるか） ・失効文書や対象外システムの文書が混入していないか
第2層　根拠	・回答が検索結果に基づいているか（渡した文書の記述と一致するか） ・存在しない仕様を生成していないか　・参照元が明示されているか
第3層　準拠	・社内ルールに準拠したコードになっているか（命名・例外処理・ログ） ・レビューで指摘なく採用できる水準に達しているか

評価の運用

評価用質問セットを作成する
現場で実際に発生する依頼を30〜50件収集し、期待される参照文書と回答をあらかじめ定義する。

同一セットで継続測定する
設定変更のたびに同じ質問で再測定し、改善が退行を招いていないかを確認する。

開発者フィードバックを収集する
回答単位で「役立った／誤り」を記録し、誤答の原因を第1〜3層のいずれかへ分類する。

失敗層に応じて打ち手を変える
第1層＝データ・検索設計、第2層＝利用ルール、第3層＝規約文書の整備。

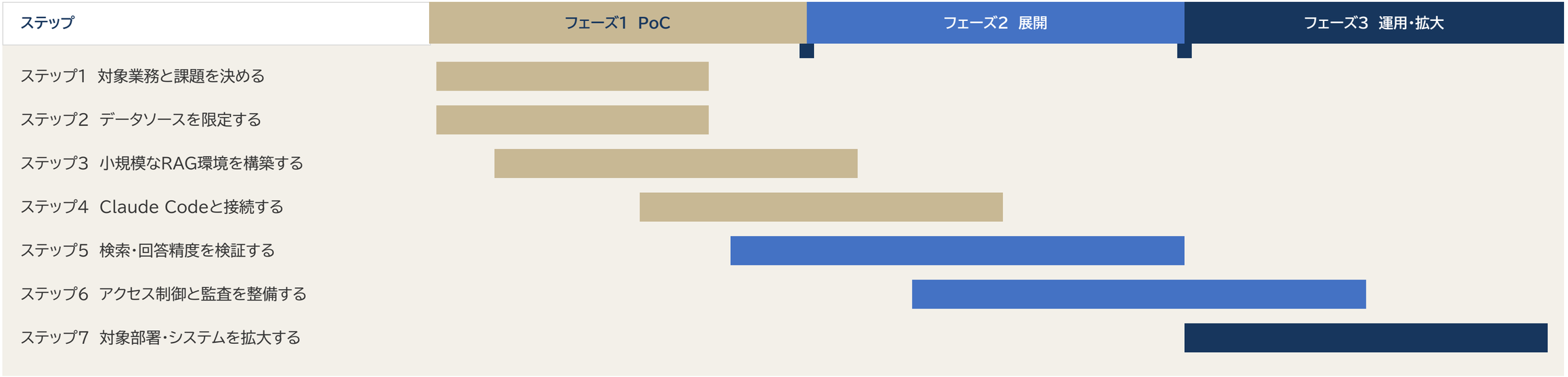
評価設計の要点

「回答が不正確」という報告だけでは改善に着手できない。**検索で失敗したのか、根拠を無視したのか、規約が文書化されていなかったのか**を切り分けること。
評価データの整備は導入後ではなく、PoC開始と同時に着手すべき作業である。

事実：構成案に記載の評価7項目／提案：3層モデルへの整理、質問セット30〜50件（仮定値・自社の業務量に応じて調整）

段階的な導入ロードマップ

全社一斉ではなく、対象業務とデータを限定したPoCから開始する。精度検証と統制整備を完了させた後に拡大することで、投資判断の根拠を段階的に積み上げる。



PoCで確認すべきKPI（目標値は仮定値／自社の現状値をベースラインとして設定すること）

調査時間

仕様・実装の調査に要する時間。
測定方法：対象業務の作業ログ／申告値
目標：短縮（仮定値）

コードレビュー時間

1件あたりのレビュー所要時間。
測定方法：レビュー管理ツールの記録
目標：短縮（仮定値）

手戻り件数

規約違反・仕様誤認による差戻し件数。
測定方法：レビュー指摘の分類集計
目標：減少（仮定値）

回答正解率

評価用質問セット（P15）に対する正答率。
測定方法：定義済み質問での定期測定
目標：向上（仮定値）

利用率

対象メンバーのうち継続利用している割合。
測定方法：利用ログの実績集計
目標：定着（仮定値）

事実：構成案に記載の7ステップ・5KPI／提案：3フェーズ区分とマイルストーン設定（提案）／仮説：KPI目標値は未設定（仮定値・要ベースライン測定）

まとめ・導入チェックリスト

成否を決めるのはデータ量ではなく、情報源の品質・アクセス権限・更新管理の3点である。小さなユースケースから着手し、評価しながら段階的に拡張することを推奨する。

現状

Claude Code単体では社内固有の知識を十分に活用できず、回答は一般論に留まる。補完は担当者の手作業に依存している。

重要課題

ナレッジは不足ではなく散在している。正式な情報源の定義・鮮度管理・権限整理が未整備なままではAI活用は成立しない。

推奨施策

RAGにより設計書・規約・障害履歴を開発作業へ組み込む。権限を引き継いだ検索と出典提示を必須要件として設計する。

期待効果

調査時間とレビュー工数の削減、手戻りの抑制、チーム間の品質差の縮小。効果は評価データにより継続的に検証する。

導入前チェックリスト

- ☐ 利用目的と対象業務が決まっている
- ☐ 正式な情報源が定義されている
- ☐ 文書の管理責任者が決まっている
- ☐ アクセス権限を検索結果へ反映できる
- ☐ 回答品質を測定する評価データがある
- ☐ 誤回答時の確認・承認ルールがある
- ☐ 利用ログと改善の仕組みが用意されている

チェックが埋まらない場合の着手順

1. 技術構築より先に「正式な情報源の定義」と「文書オーナーの任命」を行う。ここが未整備のままでは精度は上がらない。
2. アクセス権限の反映方式を情報セキュリティ部門と合意する。後付けでの設計変更は影響が大きい。
3. 評価用質問セットをPoC開始と同時に整備する。効果を語る根拠がなければ拡大の判断ができない。

ご検討いただきたい意思決定事項

① PoC対象業務・対象システムの決定

② 初期連携データと文書オーナーの任命

③ 権限・監査要件のセキュリティ部門合意

④ 評価KPIとベースライン測定の実施

事実：構成案に記載のまとめ4点・チェックリスト7項目／提案：着手順および意思決定事項4項目（提案）

お問い合わせ

本資料の内容に関するご質問、自社環境でのRAG構築・PoC実施のご相談は、下記までお問い合わせください。現状のナレッジ整備状況の確認から、対象業務の選定までご支援します。

お問い合わせ先

会社名	**Start Challenge Consulting**
メールアドレス	**info@startchg.com**
担当	**平出**

お問い合わせの際は、対象システム・想定ユースケース・現在の課題を併せてご記載いただくと、より具体的なご提案が可能です。

Start Challenge
Consulting co., ltd.
挑む力が、世界を照らす

生成AI活用・開発プロセス改善に関する
コンサルティングを提供しています。

ご相談いただける内容の例

現状評価
社内ナレッジの棚卸しと、正式な情報源・更新管理体制の整備状況の確認。

PoC設計
対象業務・連携データの選定、評価用質問セットとKPIの設計支援。

統制設計
アクセス制御・監査ログ要件の整理と、関係部門との調整支援。

Confidential 本資料の無断複製・二次配布を禁じます