～Bplats 様に AI 駆動開発を全面導入したプロジェクトの舞台裏～

AI 駆動開発支援基盤 AMANEX のご紹介

AMANEX

あまねく、たどりつく。
Everywhere, everything, within reach.

2026年8月 Koozyt, Inc

クウジット (Koozyt) について

未来のライフスタイルからあるべき形を想像する
共創型テクノロジー企業へ

空と実をつなぐクウジットです

2007 Sony CSL

創業メンバーは
ソニーCSL出身

- Wi-Fi測位サービス PlaceEngine
- CyberCode AR技術
- 笑顔認識
- 因果情報分析 CALC

2007 Koozyt

- ソニー CSL初スピンアウト
- 持分適用
- ソニーCSLの研究シーズのビジネス推進企業として独立

2011 電通総研 / INNOLAB

- ISID（現電通総研）
- 持分適用
- 資本、業務提携

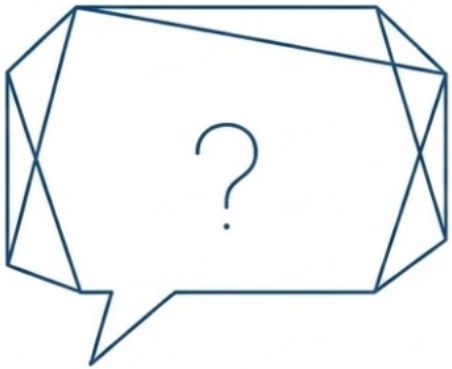
2024 Japan Deep Learning Association
日本ディープラニング協会
正会員

2026 AI導入支援コンソーシアム
AI導入支援コンソーシアム
発起起業・正会員

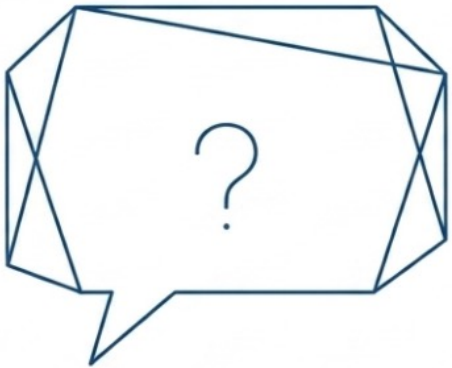
DCON
全国高専ディープラーニングコンテスト審査員

商号	クウジット株式会社（Koozyt, Inc.）
本社所在地	〒105-0014 東京都港区芝2-30-11芝コトブキビル601 https://www.koozyt.com/
設立	2007年7月3日
資本金	5,000万円
共同創業者	代表取締役社長 CEO 末吉 隆彦
共同創業者	取締役副社長 COO 塩野崎 敦
共同創業者	取締役(非常勤) 技術顧問 層本 純一

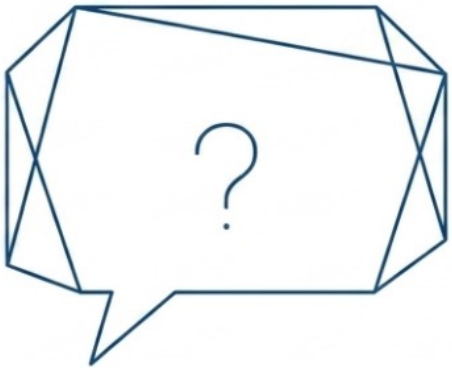
AI がコーディングしてくれる時代における “誤解”



「AIがコードを書くなら、ドキュメントはもうなくても困らない」



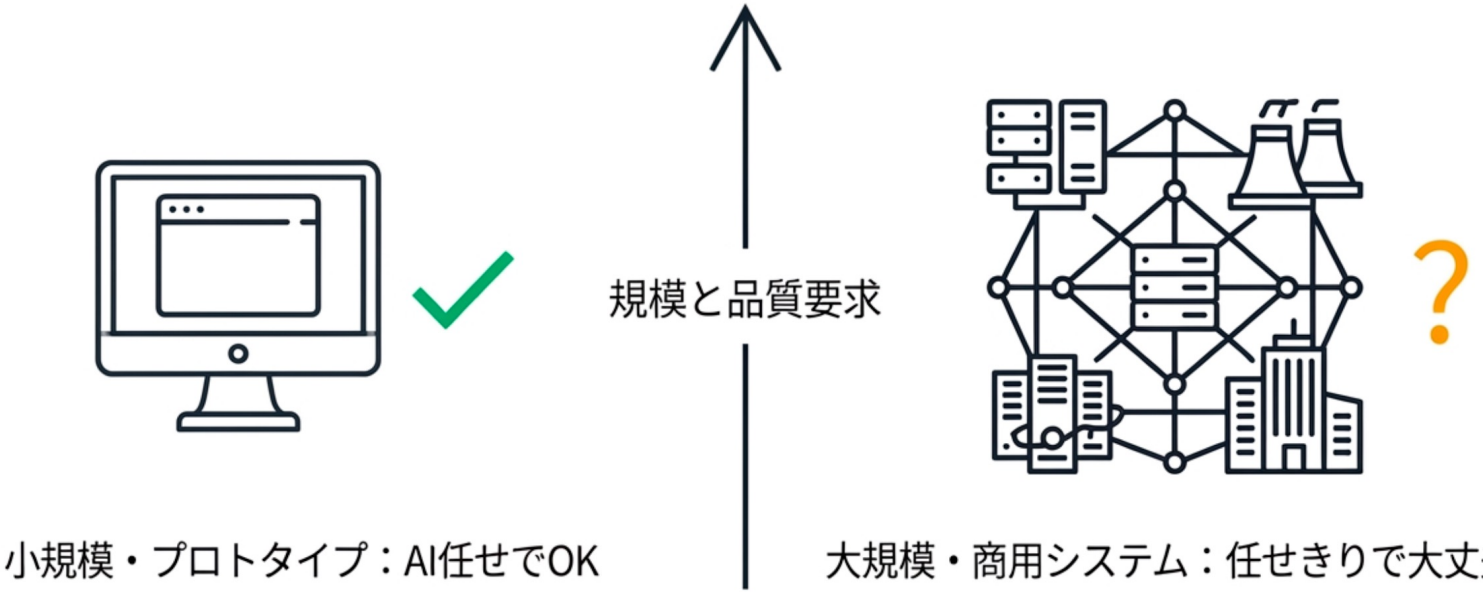
「分からないことは、その都度 AI に聞けば答えてくれる」



「必要ならドキュメントも AI が作ってくれるから問題ない」

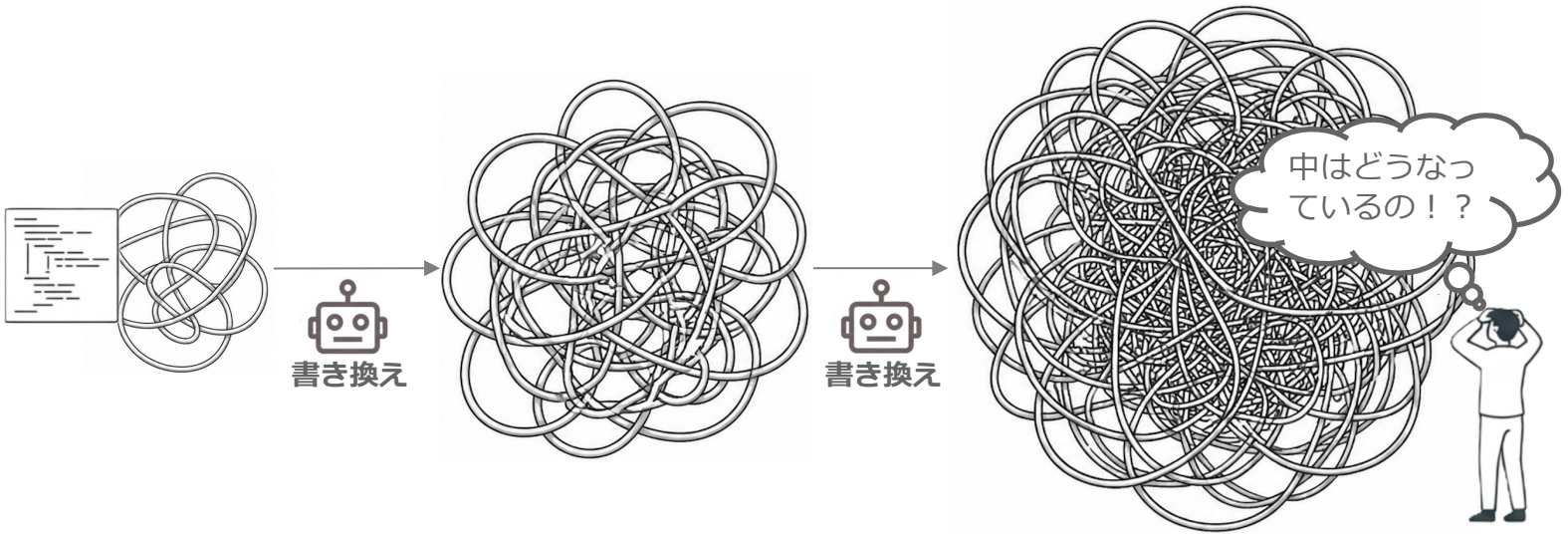
はたして本当にそうなのでしょうか？

大規模な商用システムでは話が変わる



- 小規模プロトタイプ／社内ツールなら、AI 任せで十分速く作れます
- 顧客に対する信頼と継続運用が求められる大規模商用システムを AI だけに任せると問題が顕在化します

AI コーディングは短期間で巨大なブラックボックスができあがる



別セッションのAIが継ぎ足すたび、構造の一貫性が崩れ、より複雑でスパゲッティ化していく

- AI は大量のコードを素早く生成しますが、設計や依存関係は誰の頭にも残らないまま積み上がります
- 大きなシステムでは、かえってレビューや保守コストが膨らみます

「分からなければ都度AIに聞く」にも限界がある

探索の限界

巨大コードの全体を毎回読ませることはできず、**浅い探索**で答えてしまう

ハルシネーション

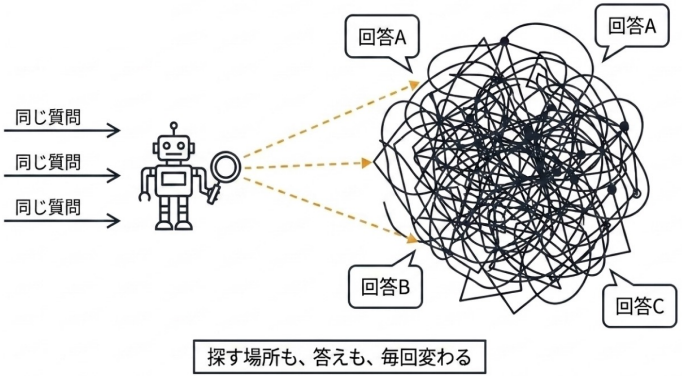
誤りが混じっても、どこにどのような誤りがあるかの**検出が難しい**

非決定性

同じ質問でも、毎回探す場所も**答えも変わりうる**

コスト

問い合わせのたびに巨大コードを読み直すのは、**時間もトークンも高くつく**

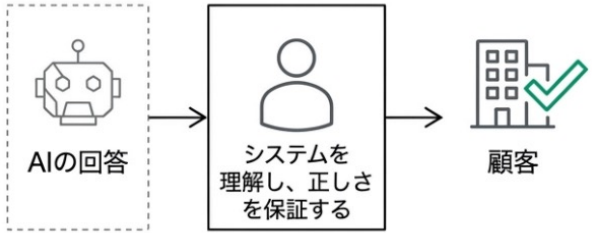


「分からないことは、
その都度 AI に聞けば
答えてくれる」

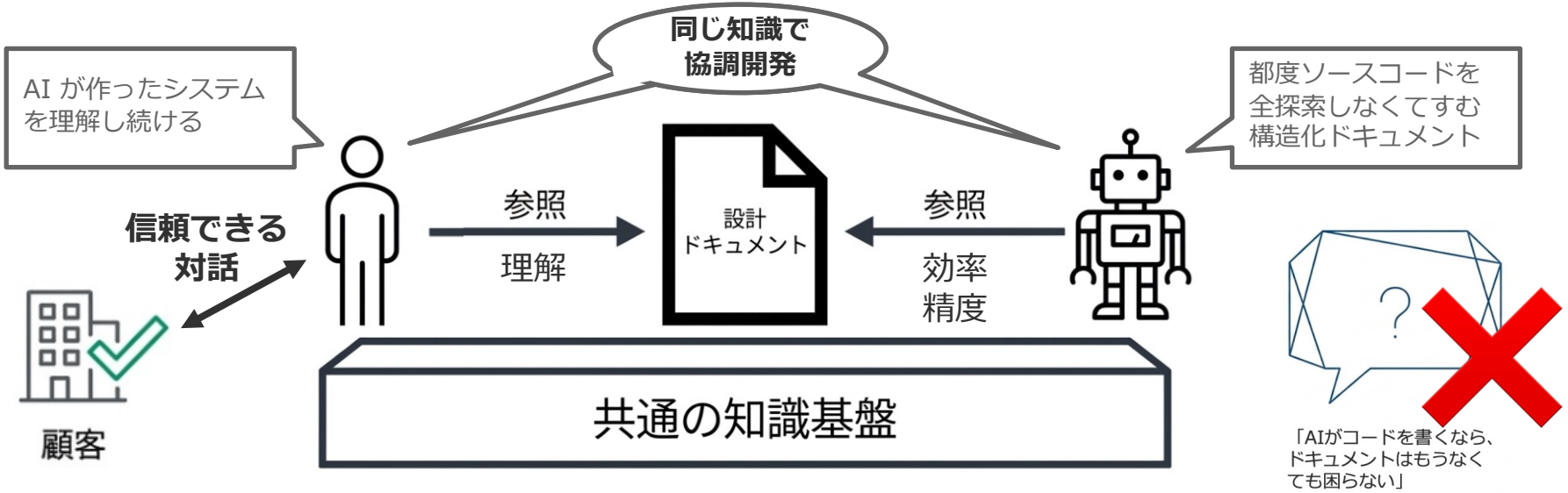
その回答を誰が保証するのか ― 信頼は精度だけでは決まらない

顧客からの問い合わせに、誰が回答内容を保証するのでしょうか？
興味深い論文があります

- 医療の実験では診断精度が同じでも、被験者はAIでなく人間の医師を選びました（Longoni et al., 2019）― 人にはこの心理的バイアスがあります
- 「AIを使った」と明かすだけで信頼が下がる、という研究報告もあります（Schilke & Reimann, 2025）― AIの回答を “AIのまま” 顧客に返すと信頼低下を招く恐れ
- だから、間に人が立って回答を保証すべき ― そのためには、人が AI の回答を理解し、正しさを確かめられなければならない

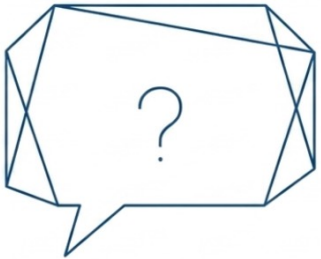


人と AI が共有する“共通の知識基盤”、それが設計ドキュメント



- 人と AI が同じ知識を参照し、同じ知識ベースで対話し、役割分担しながら協調して開発を進めます
- AI コーディング時代だからこそ
  「コード → ドキュメント化 → 知識の共有 → AI との協調開発」が重要になります

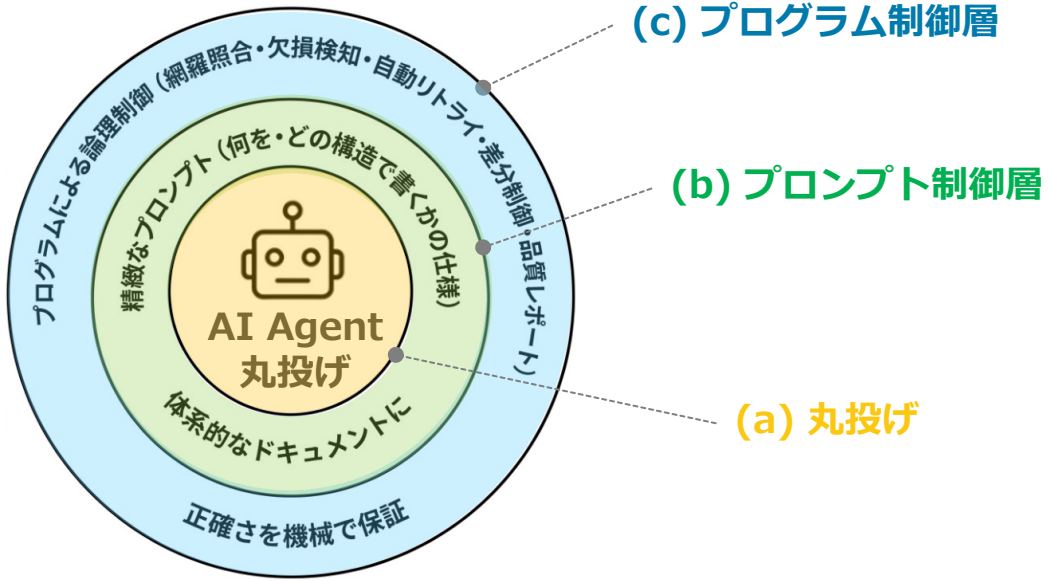
では、そのドキュメントは AIに丸投げで作れないのか



「必要ならドキュメントも AI が作ってくれるから問題ない」

本当？

実験

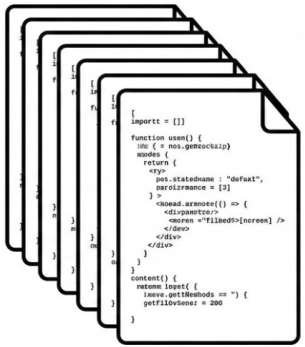


次の3レベルで実験

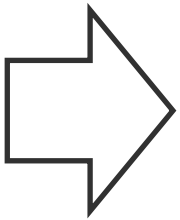
※(a)(b)は同一の AI Agent・同一モデル（Claude Code / Opus 4.8）で実施

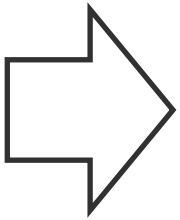
(a) AI Agent に丸投げする
(b) 精緻なプロンプトを与える（プロンプト設計）
(c) プログラム制御によるハーネス設計を行って AI Agent の外側から不具合検知とリトライ制御を行う

実験 (a): AI 丸投げは 全体“概要”をいくつか作成して終了



大規模ソースコード群
(PHP 8,000ファイル規模)





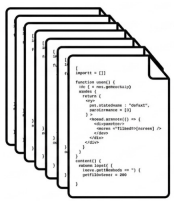
実際のプロンプト

このフォルダのソースコードを読んで、開発者向けの技術ドキュメントを作成してください。
ソースルート: src
出力先: docs
各ソースファイルと各フォルダの説明、および全体の目次（README）を作ってください。

- ファイル単位の網羅ゼロ・相互参照ゼロ、ファイル横断的記載ゼロ
- 概要ドキュメントが6つできただけでした

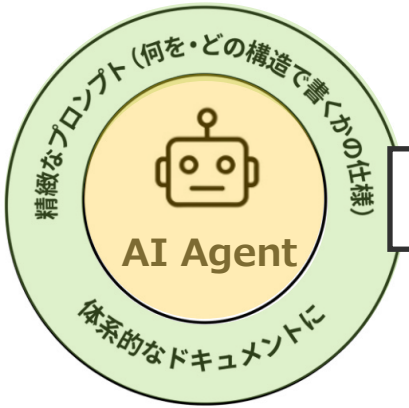
AI Agent に丸投げでは実用的な設計仕様書は作成できませんでした

実験 (b): 体系的な生成には成功 — しかし差分更新で致命的な失敗



大規模ソースコード群
および初回生成ドキュメント
(PHP 8,000ファイル規模)

新バージョンのファイル更新
（約1,050ファイルの更新が必要）



精緻なプロンプトの内容概要

①何を作るか（ファイル単位・モジュール単位・横断集約・索引という成果物一式の定義）

②どの構造で書くか（ドキュメントの詳細なフォーマット仕様 約30KB）

③何を書いてよいか（推測での創作禁止・主要要素の網羅・依存関係は参照元の位置まで明記）

④補助ツール（Serena MCP コード構造解析ツールの活用）

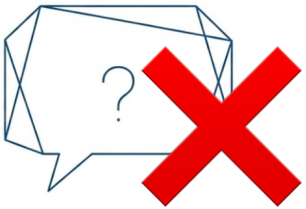
- 精緻なプロンプトを与えると、初回は体系的なドキュメント一式の生成に成功（内容もソースと突き合わせて監査し、専用ツールと互角の品質）
- しかし差分更新では：約1,050ファイルの更新が必要な新バージョンを投入したところ、244ファイル（約4件に1件）が“黙って”未反映でした
- 変更部分に対応したドキュメントを漏れなく更新するのは、精緻なプロンプトでも困難なようです

AI まかせの弱点は“失敗が見えない”こと

できました！といいながら黙って漏れ落ちが起きる

ファイル欠落だけでなく、ファイル内の項目が欠落することもあります

どこが失敗しているかわからないため結果としてかえってレビューコストを増大させてしまいます

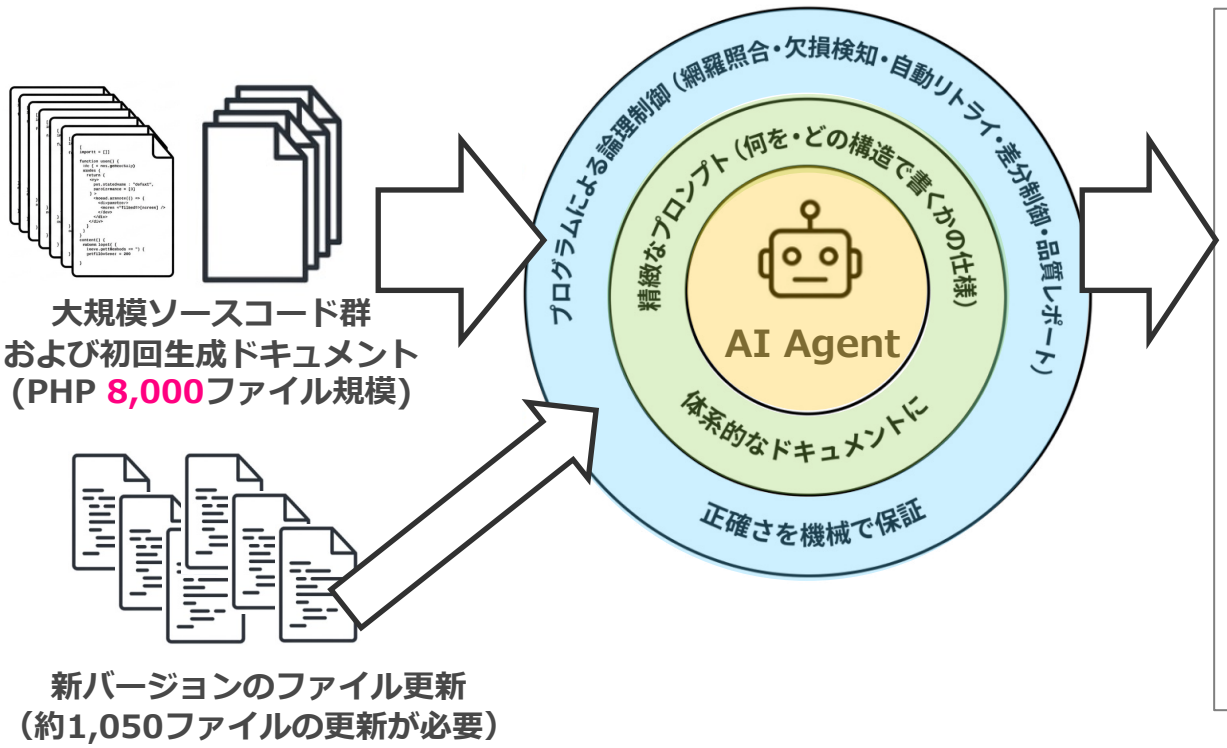


「必要ならドキュメントも AI が作ってくれるから問題ない」

- 処理しなかったファイルがあっても黙って完了になります
- 同じ入力でも毎回結果が変わり得ます（run1 ≠ run2）
- そのため機械的な検査をしないかぎり誰も気づきません ― ドキュメントが静かに劣化していきます

実験 (c): 正しいものができるまでリトライして完全なものを生成

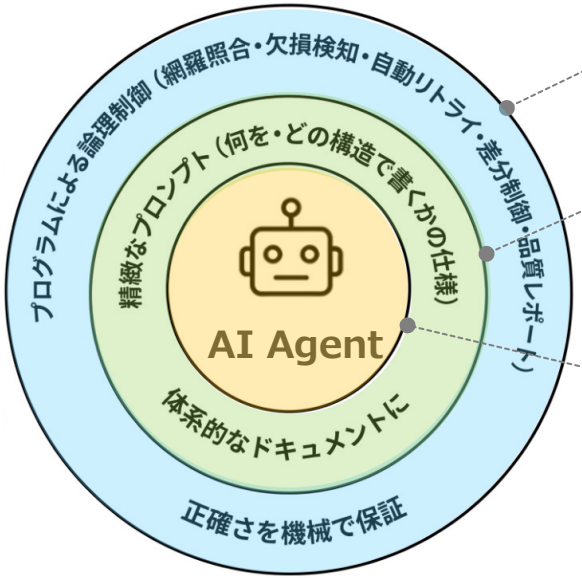
網羅性のチェック ／ 欠損検知 ／ 自動リトライ ／ 未生成箇所の明示



- 更新が必要な約1,050ファイルを漏れなく反映し、据置の約7,000ファイルはいっさい触らない — 対象すべてで抜け落ちなしでした
- 各ファイルの内容も抜け落ちなしでした
- 万一リトライ上限まで失敗しても、そのファイルをログに記録（”失敗の見える化”）します

AI + 機械的なチェック＆リトライの重要さを実測値で示しました

AMANEX docgen とは



この3層設計をまとめてパッケージ化したもの

(c) プログラム制御層

(b) プロンプト制御層

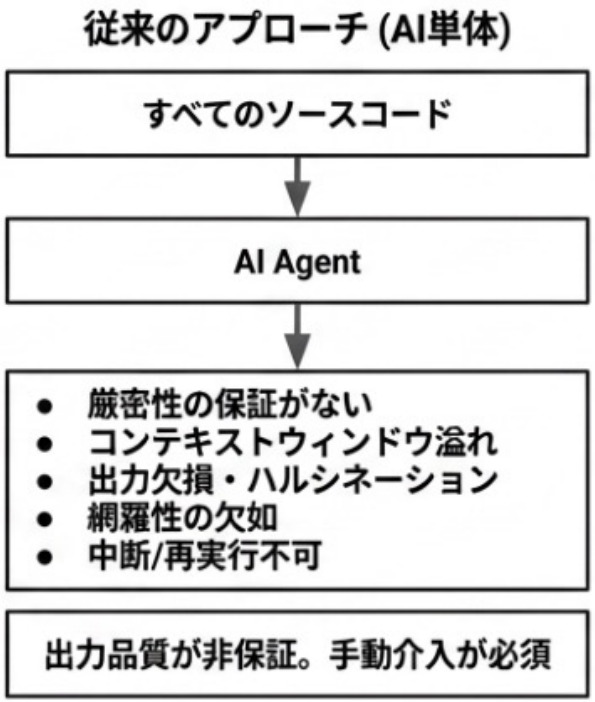
AMANEX docgen

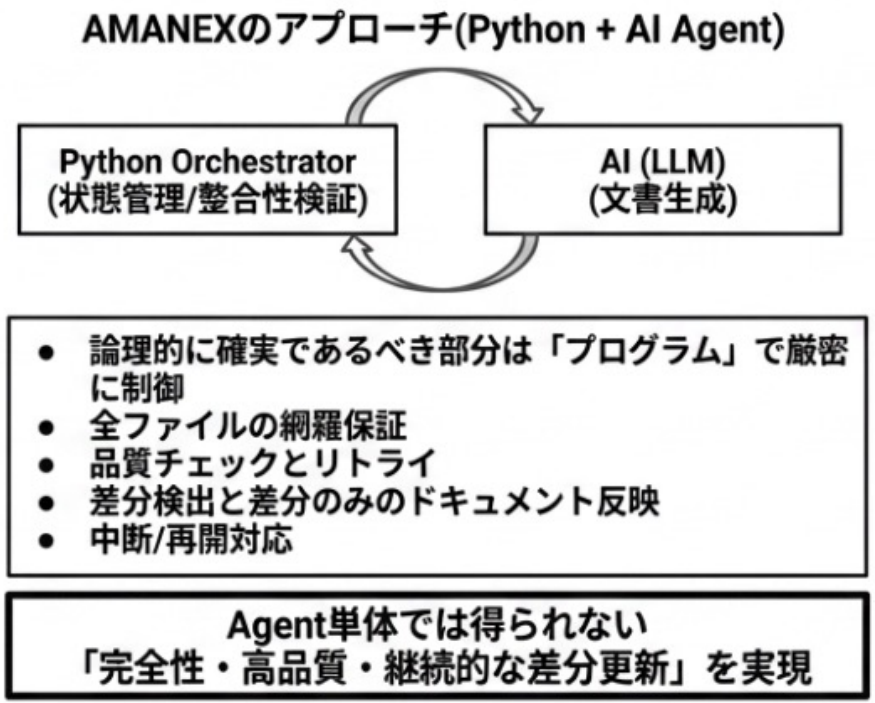
(a) AI Agent (現時点では Claude Code)

ドキュメント生成対応言語

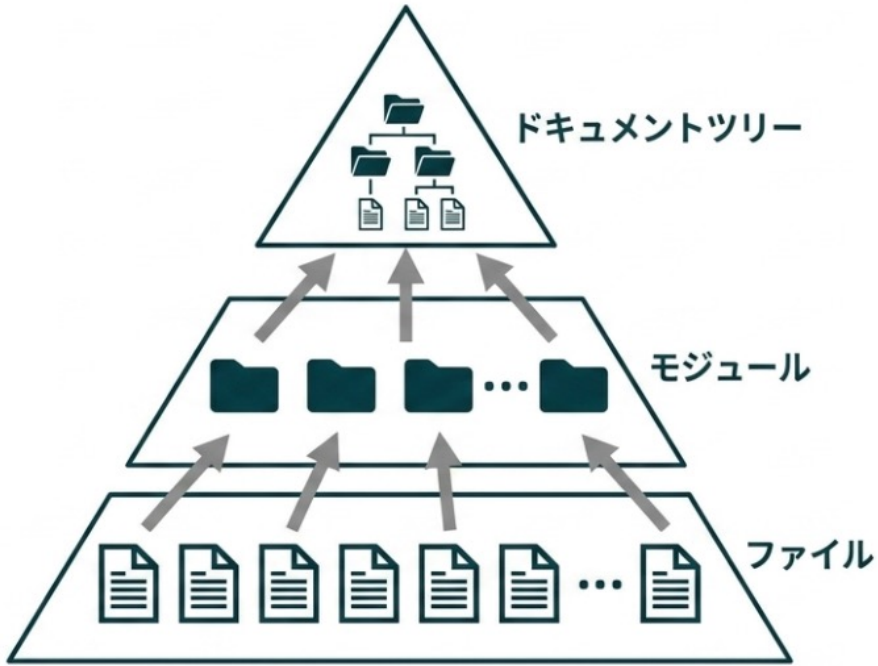
JavaScript, TypeScript, Python, PHP,　C/C++, C#, Ruby, Go, Rust など
ビジネスでの使用率調査で上位に出てくる言語を中心に対応済み
今後も拡充予定

AMANEX docgen のアプローチ ～ AI Agent に丸投げしない



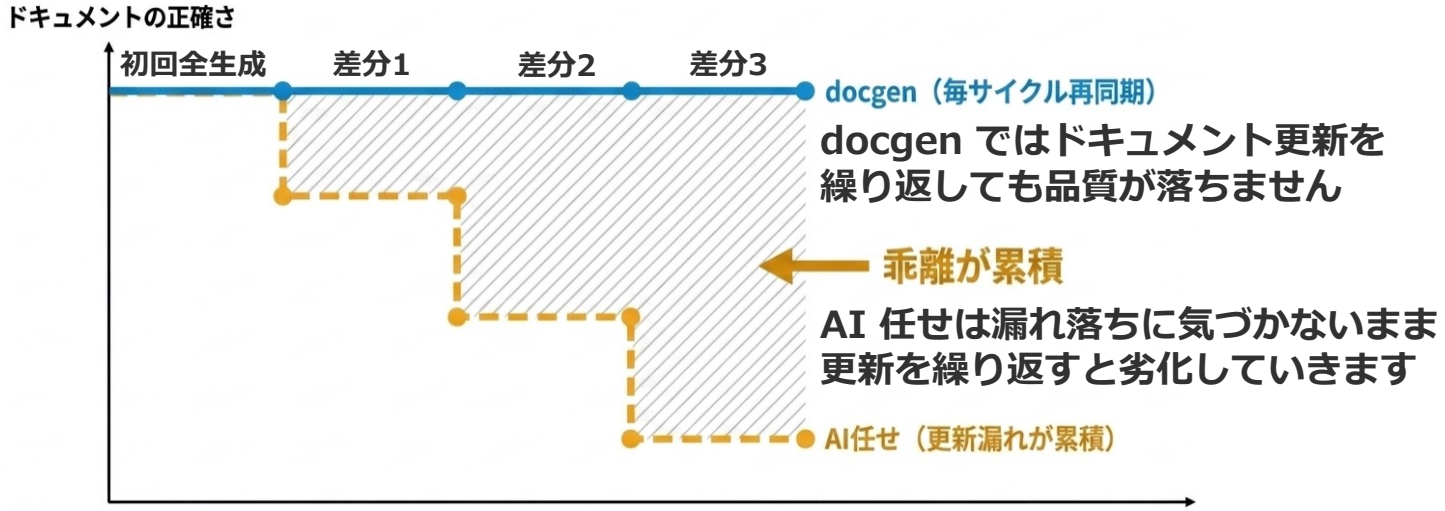


AMANEX docgen のアプローチ ～ ボトムアップビルド



- 最下層ファイル → 上位モジュールへ段階的に解析・集約 — 巨大コードでも漏れ落ちなく完遂し、完成後はトップダウンで辿れるドキュメントツリーになります
- ER図・DB定義書・モジュール依存関係・セキュリティ/不具合懸念など、エンジニアが必要とする図解や一覧も出力します（自動で HTML 化しブラウザ閲覧）
- コード構造は Serena（定義・参照・継承の解析: LSP）で正確に把握します

AMANEX docgen のアプローチ ～ ドキュメントの差分更新



- 変更箇所に対応する部分だけを置き換え、変わらない部分は触らない — git 管理でも差分が汚れません
- 読む範囲・書く範囲が最小なので、更新コストも最小
- 毎サイクルでチェック＆リトライが動くため、更新を繰り返しても品質が落ちません — AI 任せは漏れ落ちに気づかないまま劣化していきます

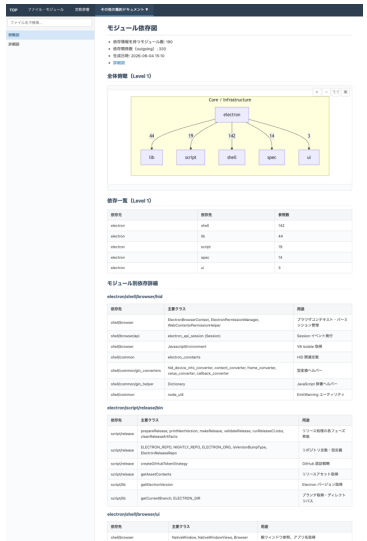
実例紹介: ～ ビープラッツ様のケース ～

対象リポジトリ：7950ファイル + 1000モジュール
生成ドキュメント数：9,083件 (75MB)

記載される内容例

ファイル／モジュールドキュメント
概要, 設計パターン, クラス/関数一覧, 依存関係, 定数一覧, データ構造, セキュリティ考慮, 技術的負債・既知の問題, 拡張性・保守性, など

ファイル横断／全体ドキュメント
定数・Enum 値 逆引き辞書, DB 定義書, 依存パッケージ一覧, ER 図, エラーコード一覧, 共通コンポーネント・ヘルパー関数一覧, モジュール依存図, ユニットテスト一覧・機能対応表, など



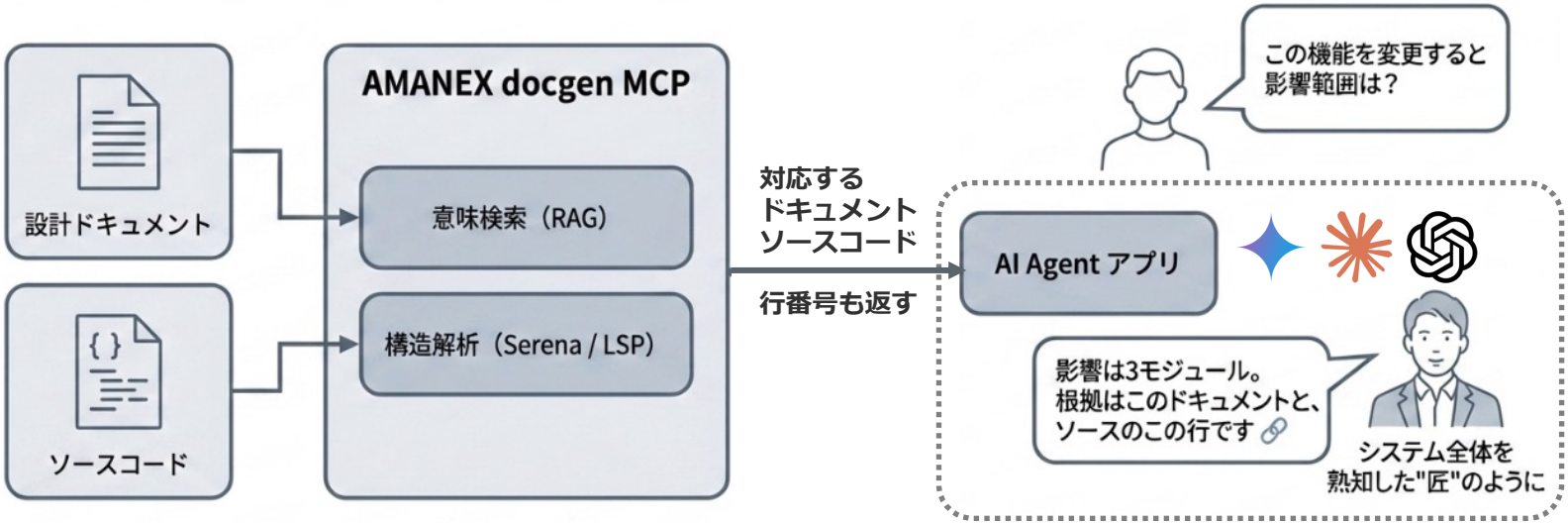
出力例（※ 実際のビープラッツ様のドキュメントでは有りません）

AI 導入支援コンソーシアム会員ページにて
ホワイトペーパー公開中

【デモ①】
ブラウザによるドキュメント閲覧例
某個人向け資産管理アプリ
PHP, JavaScript, Vue 合計1,800ファイル（約30万行）

※画面イメージは今回のデモとは別の Monolog の分析結果です

生成ドキュメントとソースコードが、“対話できる匠” になる

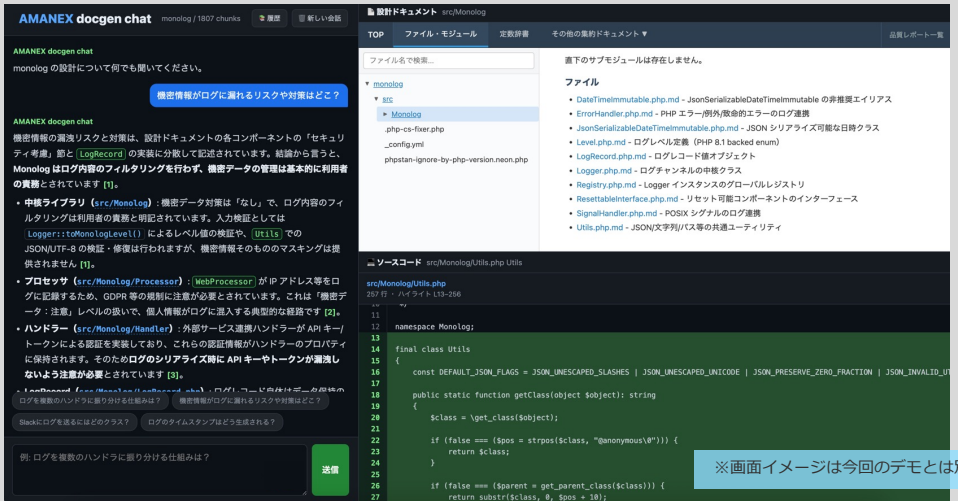


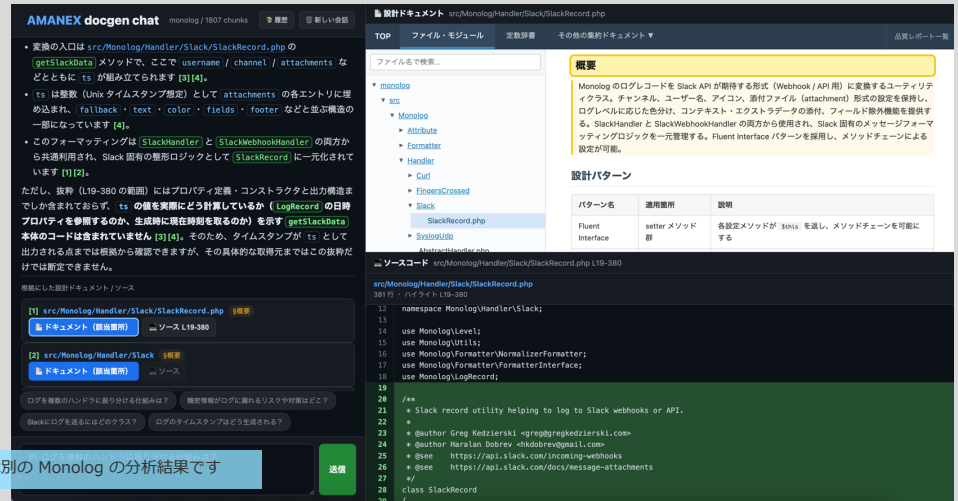
- 生成ドキュメントは、人が読むだけでなく AI の“入力”として活用できます（浅い探索を防ぎ、仕様に従った安全なコード変更を支える）
- 設計ドキュメント: 意味検索、ソースコード: 構造解析で索引化し、AMANEX docgen MCP が束ねます。MCP なので既存の AI Agent プログラムと連携できます。
- 手持ちの AI Agent から接続して質問すると、関連ドキュメントと該当ソースの場所を根拠として返す ― システムを熟知した匠・ベテランPMと対話するように使えます

Koozyt

【デモ②】
ドキュメントとソースコードを
把握した“匠 AI”と対話して設計する

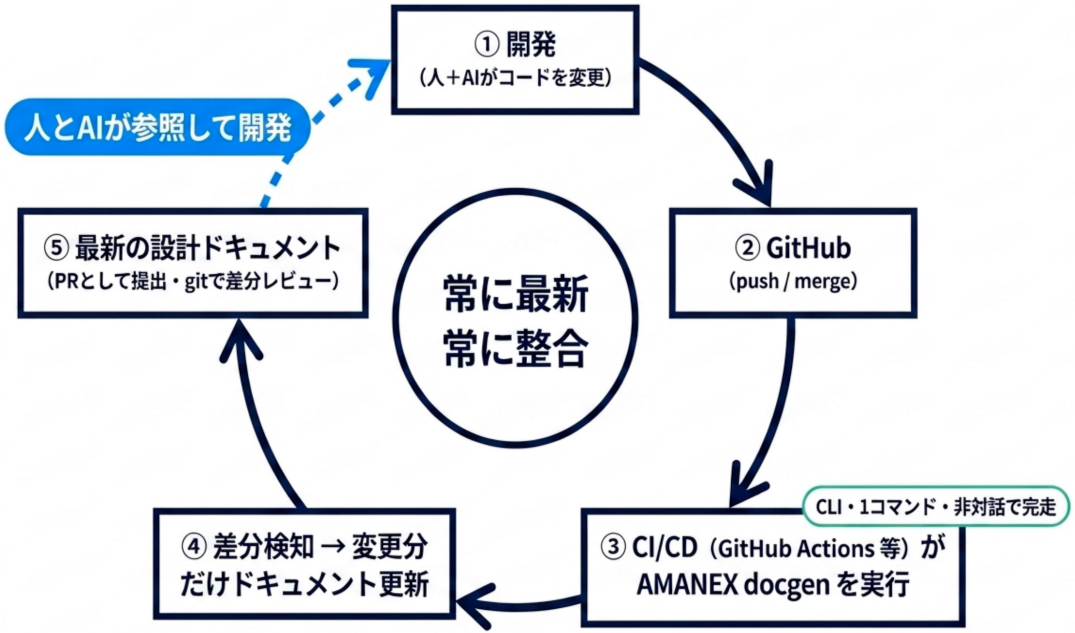
デモ①の個人向け資産管理アプリ
PHP, JavaScript, Vue 合計1,800ファイル（約30万行）
生成ドキュメント 2,340ファイル（約38万行）





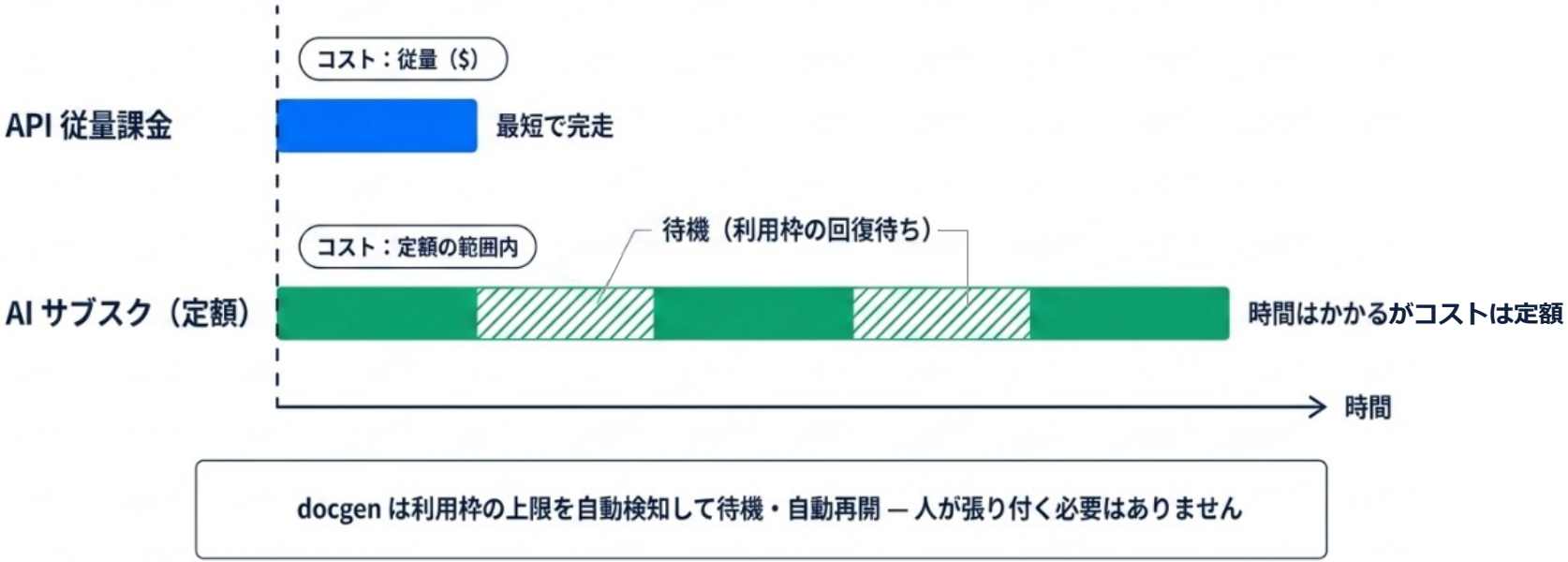
※画面イメージは今回のデモとは別の Monolog の分析結果です

AMANEX docgen を AI 駆動開発サイクルに載せる



- GitHub 連携：リポジトリ URL を指定するだけで初回生成・差分更新を直接実行（ブランチ/タグ指定可）
- CLI ツール（1コマンド・非対話・自動再開）なので、既存の CI/CD や独自の開発ループに組み込みやすい

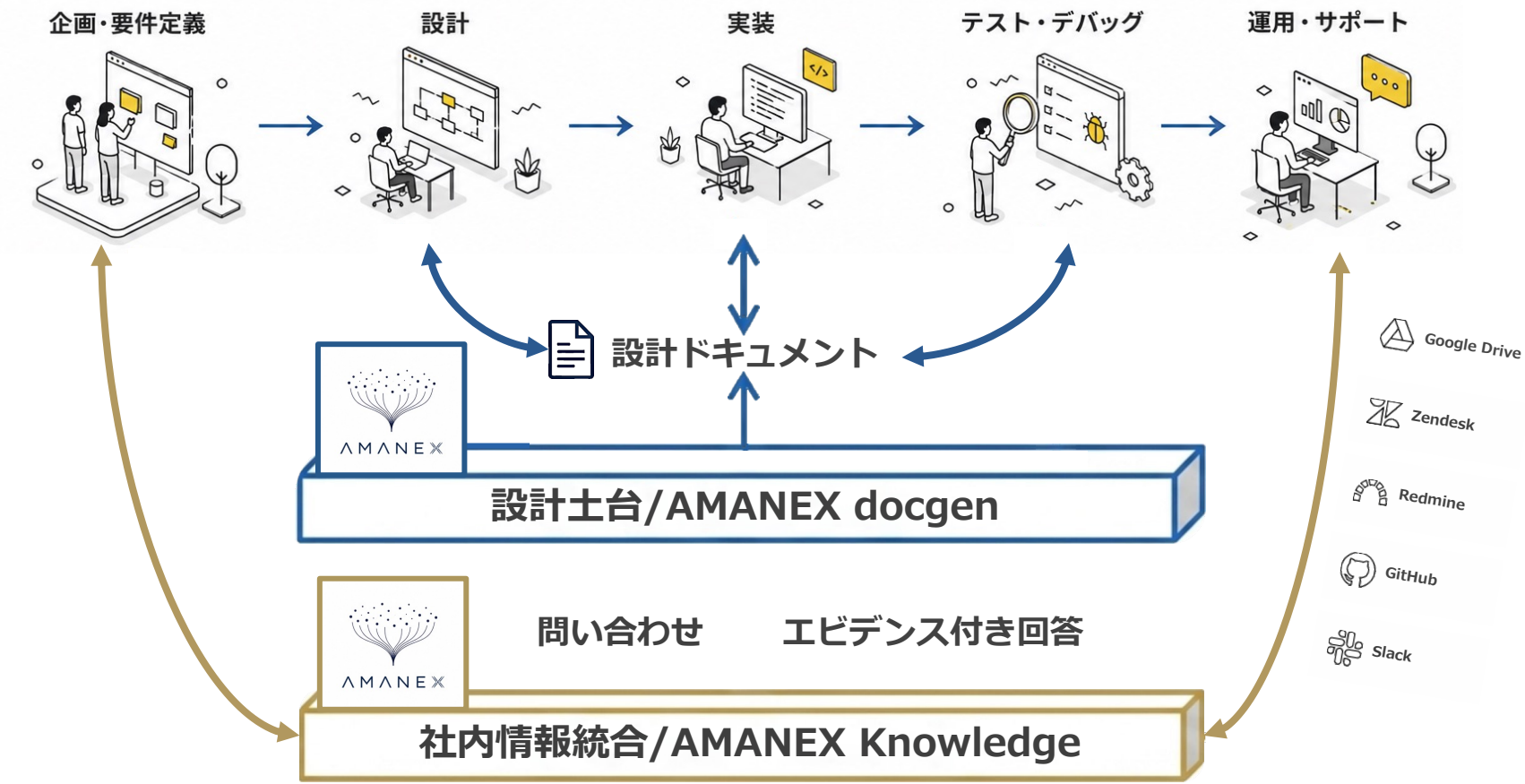
運用時の AI コストの目安 — API 従量課金とサブスク



AMANEX docgen は現時点では Anthropic社の Claude Code を使用します

- 初回生成：bplats さんの 約8,000ファイルの商用システム実績で $1,800〜$2,500 程度(従量課金換算)
- 差分更新：1ファイルあたり $0.1〜$0.5（目安 $0.2〜$0.3）(従量課金換算)（言語・変更量で変動）
- AI サブスク利用時は定額内でコストを抑えられる一方、利用枠の待機で総所要時間は伸びます

AI駆動開発の全フェーズを支える統合インフラストラクチャの土台へ



生成は AI に、保証は仕組みに、理解は人に

その役割分担を支える ― AMANEX

あまねく、たどりつく。
Everywhere, everything, within reach.