

生成AIで変える ASTERIA開発 ～ 社内ルール準拠を実現する開発支援への取り組み ～

2026年10月29日

パナソニック デジタル株式会社

A I ・ データ ・ R & D 担当 R & D 統括部 R & D 推進部

谷田 実桜



谷田 実桜 | MIO TANIDA

所属

パナソニック デジタル株式会社

A I ・ データ ・ R & D 担当 R & D 統括部 R & D 推進部

経歴

入社3年目

ASTERIAの開発経験は約2年、まだまだ勉強中です

担当業務

ID管理システムの運用・保守 / 一部でASTERIA開発を担当

所属部署について | A I ・ データ ・ R & D 担当 R & D 統括部 R & D 推進部

社内のシステム開発者に向けて開発基盤を提供しており、バッチ処理の基盤としてASTERIA Warpを全社に展開しています。基盤の提供だけでなく、命名規約や設計標準といった開発ルールの整備も担っています。

A. 背景 / ロジック理解に時間がかかる

B. アプローチ / 社内ルールをAIに渡す

C. 活用事例 / 仕様書もレビューも、社内標準どおりに

D. 結果と展望 / 現在地と、これから

A. 背景

ロジック理解に時間がかかる

あなたの現場でも起きていませんか？

Q1

他の人が作った
ソースコードやフローの
理解に **1 時間以上**
かかったことはありますか？

Q2

担当システムに
最新の仕様書
はありますか？

Q3

詳しい人に聞かないと
分からなかったこと
はありますか？

私たちの部内では、この3つが日常的に起きていました。

ベテランも若手も、「ロジック調査」を負担に感じている

6

部内の ASTERIA 開発者 13 名に聞いた

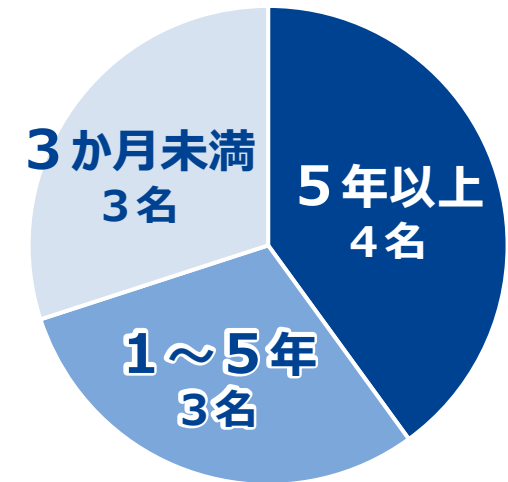
「負担が大きい作業」は経験年数に関係なく「ロジック調査」

1位 既存フローのロジック把握・調査 10名

2位 過去事例・ナレッジの探索 5名

3位 有識者への確認／問い合わせ時の根拠資料の探索／基本操作・コンポーネント仕様の理解／テスト観点・テストケースの作成／設計書・テスト仕様書の作成（同数で5項目・各3名）

「ロジック調査」と回答した
開発者の ASTERIA 経験年数



なぜロジックの把握に時間がかかるのか

※調査対象：部内の ASTERIA 開発者 13 名／

設問：現在の ASTERIA 作業で特に負担が大きい、または改善効果が大いと感じる点（複数選択可・最大3つまで）／回答数の多い順に掲載。

頼れるドキュメントがないから、毎回ゼロから読み解く

部内 ASTERIA開発者 12 名のうち 9 名

75 %

設計書・仕様書が
不足している／探しにくい

仕様書が存在しない

毎回、フローを読み解くところから始まる

仕様書が陳腐化している

改修のたびに更新されず、実態と離れていく

有識者への依存が続く

聞ける人がいる限り、属人化は解消されない

フローは動く。でも、ナレッジは残らない

※調査対象：部内のASTERIA開発者13名（任意回答12名）／設問：ASTERIA関連業務に必要な仕様書・手順書・ナレッジなどのドキュメントについて、不足している／探しにくいと感じるものを選択して下さい。（複数選択可）

As-Is

情報が分散し、
探すのも作るのも大変



To-Be

情報が集約され、
誰でも同じ品質で開発できる

探す

仕様書が不足／陳腐化
情報が分散している

→ 😡 調査に時間がかかる



探す

仕様書がすべて揃い、
情報が集約されている



作る

品質にばらつきがある

→ 😡 レビューアの負担大



作る

同じ様式で作成できる



聞く

詳しい人に頼るしかない

→ 😡 特定の人に負担が集中



聞く

有識者の知識が集約され、
誰でもアクセスできる



As-Is

情報が分散し、
探すのも作るのも大変



To-Be

情報が集約され、
誰でも同じ品質で開発できる

探す

仕様書が不足／陳腐化
情報が分散している



探す

仕様書がすべて揃い、
情報が集約されている



生成AIを活用して、個人の能力に依存しない開発チームを作る

作る

→ 🤨 レビューアの負担大



作る

同じ様式で作成できる



聞く

詳しい人に頼るしかない
→ 🤨 特定の人に負担が集中



聞く

有識者の知識が集約され、
誰でもアクセスできる



B. アプローチ

社内ルールをAIに渡す

ASTERIAの知識と社内ルールを「マニュアル」としてAIに渡す

1

ASTERIAの知識を
まとめる



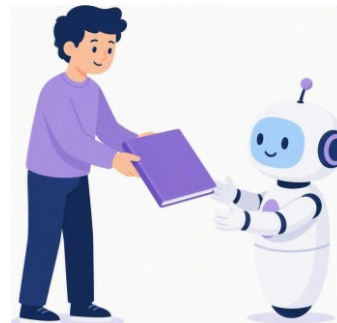
2

社内ルールを
まとめる



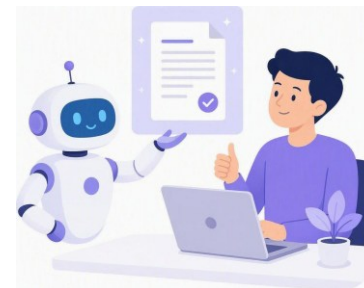
3

マニュアルとして
AIに渡す



4

それを基に
AIが答える



社内ルール×ASTERIA開発を知り尽くした「**AI**」が、開発の相棒に

足りないのは「能力」ではなく、「ASTERIAの知識」と「社内ルール」

生成 AI そのまま

- 一般的なプログラミングは得意
- × ASTERIA の設定に詳しくない
- × 社内の規約・様式を知らない

→ 毎回、前提から説明が必要



+ ASTERIAの知識 と 社内ルール

- ◎ コンポーネントを踏まえて答える
- ◎ 命名規約・設計標準に沿う
- ◎ 社内フォーマットで書き出せる

→ 前置きなしで、社内標準の開発を支援



汎用AIアシスタント



社内ASTERIA開発に特化した専門アシスタントへ

Agent Skills は、AI に知識やルールを教えるための仕組み

正体は AI 用の
「お仕事マニュアル」



1 つのフォルダに
手順やルールがまとまっている

必要なときに
AIが自分で選ぶ



指示内容から
自動でマニュアルが必要か判断

フォルダを渡すだけで
マニュアル共有



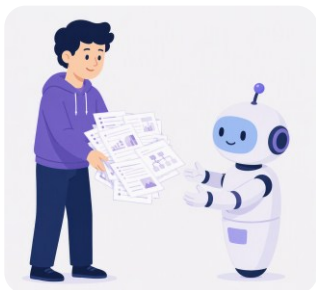
チーム内での共有が簡単

マニュアルを共有することで、誰でも 同じ品質で AI を動かせる

特別な開発は不要。社内に散らばる資料をAIに渡すだけ

1

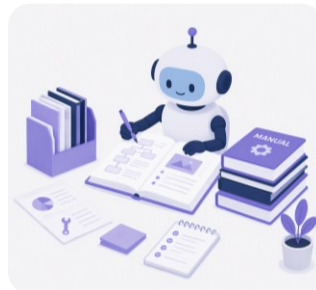
手元の資料を
そのまま渡すだけ



作る手間はゼロ

2

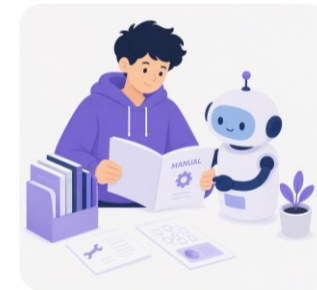
AIが
マニュアルにまとめる



まとめる手間もゼロ

3

AIが作った
マニュアルを確認する



人がやるのは
最後の確認だけ

AIが書く

マニュアルの核となる「SKILL.md」は、Markdown のテキストファイル。
AIが作るので、
自分で書く必要はありません。

追加・変更も簡単

機能を足すときも、
ルールを変えるときも、
SKILL.md を変更するだけ。

テンプレート等も同じフォルダに

仕様書テンプレートなどは
同じフォルダに置くだけ。
テンプレート自体もAIが作ってくれます。

スキルはフォルダ1つ

- asteria-master/
 - SKILL.md
 - templates/ (テンプレート)
 - 仕様書のフォーマットなど
 - references/ (詳細マニュアル)
 - 命名規約・設計標準など

SKILL.md (AIへの指示書) の中身 ※抜粋

```
---  
name: asteria-master  
description: >  
  ASTERIA Warp (アステリアワープ) に特化したスキル。ノーコードデータ連携ツールの  
  フロー設計支援、コンポーネント提案、フロー処理内容の自然言語説明、テスト仕様書の作成、  
  ...  
  このワークスペースでフロー設計やコンポーネント説明を行うときは、以下の資料を参照して回答の精度を上げるこ  
  と。  
  - `references/COMPONENTS_LIST.md`  
  - どのコンポーネント/関数が見えるかの一覧確認に使う。  
  - 設計時の候補出し (代替コンポーネント提案を含む) に使う。
```

書いてあるのは、一般的な文章。だから、専門知識がなくても読めて直せます

社内標準を知っている AI が、設計もドキュメントも支える

マニュアルに持たせたもの

知識 | ASTERIAルール

コンポーネント仕様・マニュアル

➡ ASTERIA固有の知識を補う

知識 | 社内ルール

命名規約・設計標準・
社内製コンポーネント仕様

➡ 社内標準に沿って答えさせる

ドキュメントテンプレート

仕様書やテスト仕様書のフォーマット

➡ ドキュメントの様式を統一する

マニュアルによって実現できること

1 社内ルールに準拠した支援が受けられる

社内製コンポーネントを使った 設計 の提案
プロパティ設定まで含んだ 開発手順 の提案

2 品質・粒度がそろったドキュメントができる

統一フォーマットの 仕様書 を生成
社内ルールを踏まえた テスト仕様書 を生成

やりたいことをチャットで伝えるだけ。使い道は盛りだくさん

質問・調査

「このフローは何をしている？」
に即答する

仕様書の生成

処理内容を
社内フォーマットの仕様書に

レビュー・規約チェック

社内基準・命名規約
に照らして指摘する

設計提案

要件から
社内標準に沿った設計案を出す

テスト仕様書の生成

社内のテスト観点で
テストケースを作る

改修手順書の生成

環境準備から確認までの
手順を書き出す

C. 活用事例

仕様書もレビューも、社内標準どおりに

1 質問・調査

2 フロー仕様書の生成

3 レビュー・規約チェック

4 設計提案

複雑な処理も、1分で中身を説明してくれる

プロンプト

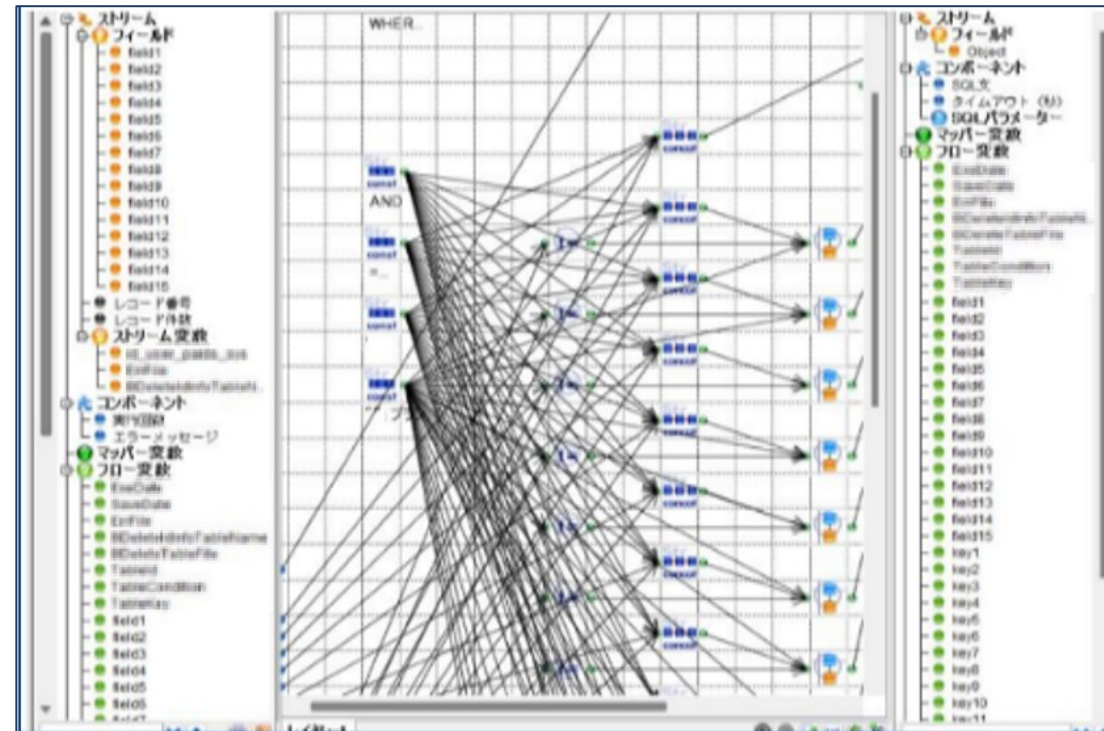
「この Mapper は何をしている？」

入力ファイル（コンテキスト）

プロジェクトファイル

パラメータファイル

+ さらに精度を上げるなら
テーブル定義書・サンプルデータ



1 質問・調査

2 フロー仕様書の生成

3 レビュー・規約チェック

4 設計提案

複雑な処理も、1分で中身を説明してくれる

出力（Mapper の解説）

- ✓ まず概要を簡潔に
- ✓ 順序立てて説明も
- ✓ サンプル例を示す
- ✓ 動かして確かめられる

ひとことで言うと

このMapperは、テーブルごとに違う「削除の条件」を毎回自動組み立てし、次のSQL実行コンポーネントへその場で渡しています。

3 SQL組み立てロジック（ステップ）

1 先頭固定文字列 + テーブル名 + WHERE

関数: `Const1` ("DELETE FROM ") → `Concatenate1` ← `FV/TableId` → `Const2` (" WHERE

```
DELETE FROM <TableId> WHERE
```

2 1つ目のキー条件は無条件で追加（必須）

`Concatenate2` = `FV/key1` + `Const4` (" = ") + `FV/field1` + `Const5` (" ")

※出力サンプルは抜粋・簡略化して掲載しています。

1 質問・調査

2 フロー仕様書の生成

3 レビュー・規約チェック

4 設計提案

複雑な処理も、1分で中身を説明してくれる

出力（Mapper の解説）

- ✓ まず概要を簡潔に
- ✓ 順序立てて説明も
- ✓ サンプル例を示す
- ✓ 動かして確かめられる

● シンプルな例（条件1つ）

対象テーブル T_MEMBER （会員マスタ）

削除条件 member_id = M0001234

```
DELETE FROM T_MEMBER
WHERE member_id = 'M0001234'
```

T_MEMBER

1 member_id

-- プリセットを選択 --

+ 条件を追加

記号 (\$?) を混ぜて試す

● 組み立てられるSQL（安全化前）

```
DELETE FROM T_MEMBER
WHERE member_id = 'M0001234'
```

※出力サンプルは抜粋・簡略化して掲載しています。

1 質問・調査

2 フロー仕様書の生成

3 レビュー・規約チェック

4 設計提案

テンプレートに従って生成。誰が作っても同じクオリティ

入力

「このフローの
仕様書を作って」

長いプロンプトも
テンプレートも
毎回説明する必要なし

出力

SampleBatch_M_DataSync01
フロー仕様書（公開サンプル） v1.0

目次
ドキュメント情報
概要
可視化
フロー構成

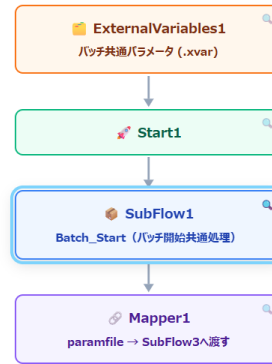
処理シーケンス
コンポーネント
サブフロー変数
SQL仕様
変数一覧
入出力仕様
エラー処理
移行チェック
制約・注意

SampleBatch_M_DataSync01

「ユーザーデータ外部連携」を担うメインフロー。
バッチ基盤共通サブフロー（Batch_Start）で開始処理を行い、パラメータ読み込み後に業務サブ
業務サブフロー内でユーザ情報をDBから取得・整形したCSVファイルを生成し、FTP転送（FTPS）
を行い EndResponse1 で終了する。

処理フロー図（SampleBatch_M_DataSync01）

各ノードをクリックすると詳細が右パネルに表示されます



凡例

Start / End
SubFlow
Mapper
External

変数マッピング
外部変数ファイル

呼び出しサブフロー一覧

コンポーネント	呼び出し先フロー	役割
SubFlow1	BatchCommon\Batch_Start	バッチ開始共通処理
SubFlow3	SampleBatch_InitSub	パラメータ読み込み
SubFlow4	SampleBatch_S_DataSync0101	業務主処理（CSV生成・FTP転送）

SubFlow1 - Batch_Start

バッチ基盤共通サブフロー（BatchCommon\Batch_Start）を呼び出す。バッチの開始ログ記録・ジョブスケジューラ連携などの共通開始処理を実施する。

★ なぜ必要か: 基盤の標準規約に従い全M系フローの冒頭で呼び出す必須処理。バッチの実行状態をシステムに登録し、異常時の通知基盤を有効化する。

「理解する」と「残す」が同じ作業に。仕様書づくりが後回しにならない

※出力サンプルは抜粋・簡略化して掲載しています。

1 質問・調査

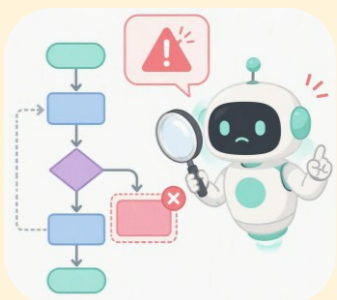
2 フロー仕様書の生成

3 レビュー・規約チェック

4 設計提案

自動でレビューされ、ルール違反はその場で警告

フロー確認のたび
社内ルールも自動チェック



指摘してほしい観点を
毎回書き出す必要はなし

禁止
コンポーネント推奨外の
設定値性能・
OOMリスク命名規約
違反

チェック項目	判定	詳細
LoopEndの出力形態	NG	初期値「ストリームをまとめる」のまま。「最後のストリーム」へ変更が必要。
フェッチ行数	NG	5,000件（上限1,000件超過）。行数の見直しが必要。
禁止コンポーネント	OK	利用禁止・非推奨コンポーネントの使用なし。

レビュー前に、自分で気づける。レビュアーの負担も軽減

※出力サンプルは抜粋・簡略化して掲載しています。

1 質問・調査

2 フロー仕様書の生成

3 レビュー・規約チェック

4 設計提案

要件を伝えるだけで、社内標準に沿った設計を提案

入力

「受注データをRDBから
取得しCSVを出力したい」

入力ファイル（コンテキスト）

テーブル定義書

命名規約も
設計標準も
毎回説明する必要なし

出力

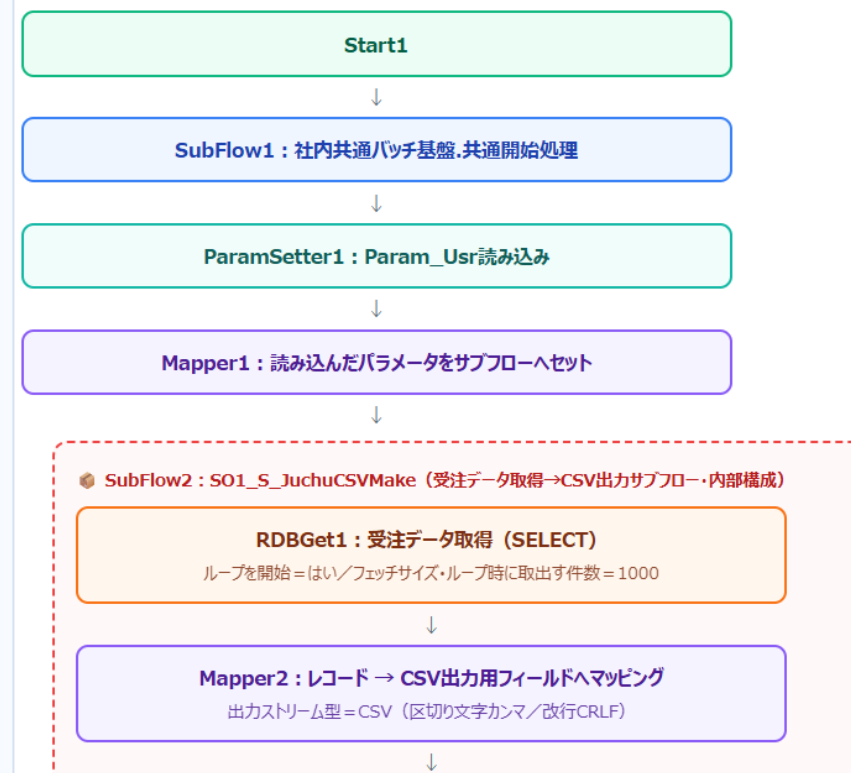
✓ フロー構成

社内標準に沿った構成を
前提説明なしで提案

✓ パラメータ設定

✓ SQL

2.1 コンポーネント構成



※出力サンプルは抜粋・簡略化して掲載しています。

1 質問・調査

2 フロー仕様書の生成

3 レビュー・規約チェック

4 設計提案

要件を伝えるだけで、社内標準に沿った設計を提案

入力

「受注データをRDBから
取得しCSVを出力したい」

入力ファイル（コンテキスト）

テーブル定義書

命名規約も
設計標準も
毎回説明する必要なし

出力

✓ フロー構成

✓ パラメータ設定

命名規約に沿った名称
一括設定できるCSVも

✓ SQL

#出力先ディレクトリ

M01_OutputDir=[WorkFile]/JuchuCSV/S01_M_JuchuCSV01/

#CSVヘッダー行

M01_HeaderLine=受注番号,受注日,得意先コード,得意先名,商

 CSVで一括設定できるテキスト（フロー変数）

フローデザイナーの「変数の追加」→「CSV インポート」タブへ、以下のテキストをそ

```
usrconf,String,[usrconf]/S01_JuchuCSV/S01_J  
sysconf,String,[sysconf]/S01_JuchuCSV/S01_J  
M01_OutputDir,String,,false,true,受注CSV出力  
M01_HeaderLine,String,,false,true,CSVヘッダ  
M01_OutputFile,String,,false,true,受注CSV出
```

※出力サンプルは抜粋・簡略化して掲載しています。

1 質問・調査

2 フロー仕様書の生成

3 レビュー・規約チェック

4 設計提案

要件を伝えるだけで、社内標準に沿った設計を提案

入力

「受注データをRDBから
取得しCSVを出力したい」

入力ファイル（コンテキスト）

テーブル定義書

命名規約も
設計標準も
毎回説明する必要なし

出力

✓ フロー構成

✓ パラメータ設定

✓ SQL

フロー内で使用する
SQLの内容も提案

SQL文

```
SELECT
  juchu_no      AS 受注番号,
  juchu_date    AS 受注日,
  tokuisaki_cd  AS 得意先コード,
  tokuisaki_mei AS 得意先名,
  shohin_cd     AS 商品コード,
  shohin_mei    AS 商品名,
  suryo         AS 数量,
FROM
  T_JUCHU
WHERE
  juchu_date = ?juchu_date?
ORDER BY
  juchu_no
```

SQLパラメーター

マッピング元

?juchu_date?

FV/M01_TargetDate（要確認：抽出対象日の決め方）

※出力サンプルは抜粋・簡略化して掲載しています。

やりたいことをチャットで伝えるだけ。使い道は盛りだくさん

質問・調査

「このフローは何をしている？」
に即答する

仕様書の生成

処理内容を
社内フォーマットの仕様書に

レビュー・規約チェック

社内基準・命名規約
に照らして指摘する

設計提案

要件から
社内標準に沿った設計案を出す

テスト仕様書の生成

社内のテスト観点で
テストケースを作る

改修手順書の生成

環境準備から確認までの
手順を書き出す

D. 結果・展望

現在地と、これから

利用者全員が効果を実感。PoC後も利用を継続

PoC実施期間

2026/07/24～8/24
(15 営業日)

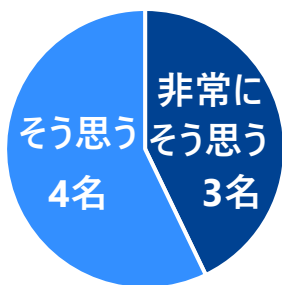
参加者

部内の ASTERIA 開発者 10 名

利用状況

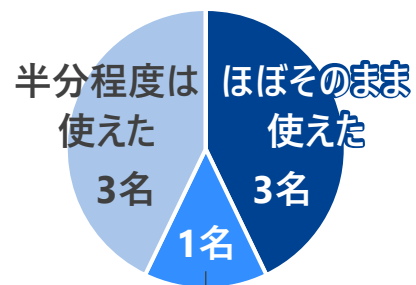
実際に利用 7 名 / 導入のみ 3 名

探す手間は減ったか



非常にそう思う	3名
そう思う	4名
どちらともいえない	0名
あまりそう思わない	0名
全くそう思わない	0名

成果物の完成度



ほぼそのまま使えた	3名
軽微な修正で使えた	1名
半分程度は使えた	3名
大幅な修正が必要だった	0名
ほとんど使えなかった	0名

作業時間は、7 件合計で 24.6 時間削減

PoC 期間中に最も効果があった作業を回答（確認・修正の時間を含む実績値）

従来のやり方 63.2 時間

Agent Skills 38.6 時間

38.9 % 時間削減

※ PoC最終アンケート（2026 年 8 月実施）より抜粋。回答は実際に利用した 7 名。

参加者の声

ロジック調査 / ASTERIA 経験 1年以上～3年未満

「フローやマップ内の複雑な処理を
解析する必要がなくなった」

社内ルール / ASTERIA 経験 3か月未満

「社内ルールに準拠した観点で、
レビューを細部まで見てくれたので助かった」

調べる手間 / ASTERIA 経験 3か月未満

「わからないことを、
人に聞かなくてもすぐに解消できる」

手戻り防止 / ASTERIA 経験 5年以上

「ワーニングの確認が
リリース時以外にもできるようになった」

これからの改善

1 「どう指示すればいいかわからない」に応える

GitHub Copilot 未経験者に多かった声



これからも **定着支援** を行い
みんなが同じ品質で開発できるチームへ

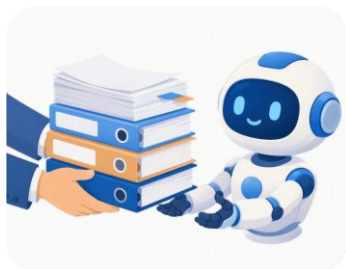
2 精度・性能の改善

- ・存在しないプロパティを挙げるなど、不具合解消
- ・エラー対処法・障害事例など、未実装の知識追加

※ 自由記述は PoC 中間・最終アンケート（2026 年 8 月実施）より抜粋。

特別な開発は不要。社内のルール文書を渡すだけ

既存文書を
「そのまま」使える



命名規約・設計標準。
分散した文書をAIがまとめる

特別な開発は
「不要」



プログラミングは不要。
文書をフォルダにまとめるだけ

1つの文書から
「小さく始める」



まずは 1 つの文書から。
効果を見ながら広げられる

フォルダを渡すだけで
「横展開」



フォルダをコピーするだけで、
チーム全員が同じ品質で使える

社内に眠っている文書が、そのまま AI の力になる

AIで進化するASTERIA開発：社内ルールを「マニュアル」として活用

やったこと（アプローチ）

手元の資料をAIに渡す



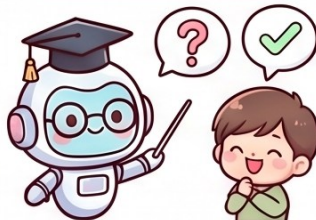
社内サイトや既存の手順書など、今ある資料をそのままAIに投入します。

AIがマニュアルにまとめる



AIが内容を整理し、AI自身が読みやすい「お仕事マニュアル」を自動作成します。

マニュアルを基にAIが回答する



学習した社内ルールに基づき、専門的な開発アドバイスや回答を行います。

できること（活用内容）

ルールに準拠した設計・実装支援



命名規約や設計標準に沿った構成案や、具体的な設定手順をAIが提案します。

ドキュメントの自動生成



複雑なフローの中身を読み解き、統一された様式の仕様書を1分で作成します。

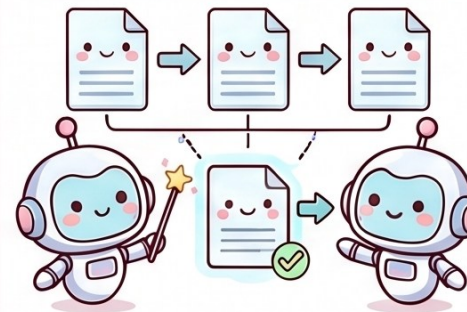
目指す成果（ゴール）

誰でも一定品質で開発できるチームへ



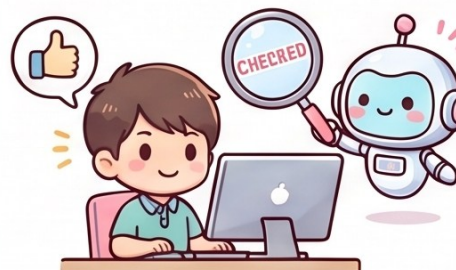
経験年数に関わらず、AIの支援で高い開発品質を維持できるようになります。

チーム全員が同じ形式のドキュメントを作成



AIがテンプレートに従うことで、資料の書き方や粒度のバラつきを解消します。

一人でも社内ルールに沿ったセルフレビューが可能



レビューアに確認する前に、AIがルール違反やリスクを自動でチェックします。

チームの知識は属人化せず永続する



特定の有識者に頼らず、ナレッジがAIを通じてチーム全体の資産として残ります。



Panasonic



 Powered by Digital