

$A^2P^2$

# AI Assisted Programming Process

An Example

Using Windsurf

酒匂寛@DDC



本日は、生成AIを開発プロセスに効果的に取り入れる「A2P2」、AI Assisted Programming Processについてご紹介します。A2P2は、丸投げ型のAI利用開発プロセスとは異なり、AIと人間が高度なインタラクションを行いながら、段階的にシステムを構築していくスタイルです。今回の事例では、「Windsurf」というAIアシスタントを中心に、買い物リストを例題として、要求定義から最終的な実装、マニュアル作成までの一連のプロセスをご覧ください。具体的なプロセスや課題解決の流れを通じて、A2P2の特徴と実践的なメリットを感じていただければと思います。



# A<sup>2</sup>P<sup>2</sup> とは

- A<sup>2</sup>P<sup>2</sup> (AI Assisted Programming Process ・ エーツーピー ツー) とは、生成 AI を開発プロセスの中に組み込んで、人間と高度なインタラクションを行いながら開発するスタイルです
  - A<sup>2</sup>P<sup>2</sup>はこのプレゼンのための造語です
- 世の中には AI に言葉を使ってほぼ「丸投げ」を行うスタイルのツールもありますが、そうしたツールはきめ細やかな意思決定や、編集作業、試行錯誤などを繰り返したい場合には不向きです
- この事例報告では A<sup>2</sup>P<sup>2</sup> ツールの一つである windsurf という AI アシスタントを利用し、開発プロセスの中に組み込まれた生成AIの一つの姿を示します

# Windsurf

The image shows a promotional banner for Windsurf AI IDE. At the top, there is a navigation bar with a logo, 'Windsurf AI', and links for '比較' (Comparison), '動画' (Videos), 'コミュニティ' (Community), 'FAQ', and a language selector '日本語'. The main content area features a large heading '開発を革新する Windsurf AI IDE' and a paragraph describing it as an intelligent development environment. A prominent blue button says 'Windsurf AI IDE を体験する'. On the right, a screenshot of the IDE interface is shown, displaying a sidebar with 'Windsurf' and 'Workspaces' sections, and a main editor area with a 'Write with Cascade' prompt and a chat window.

**Windsurf AI** 比較 動画 コミュニティ FAQ 日本語

## 開発を革新する Windsurf AI IDE

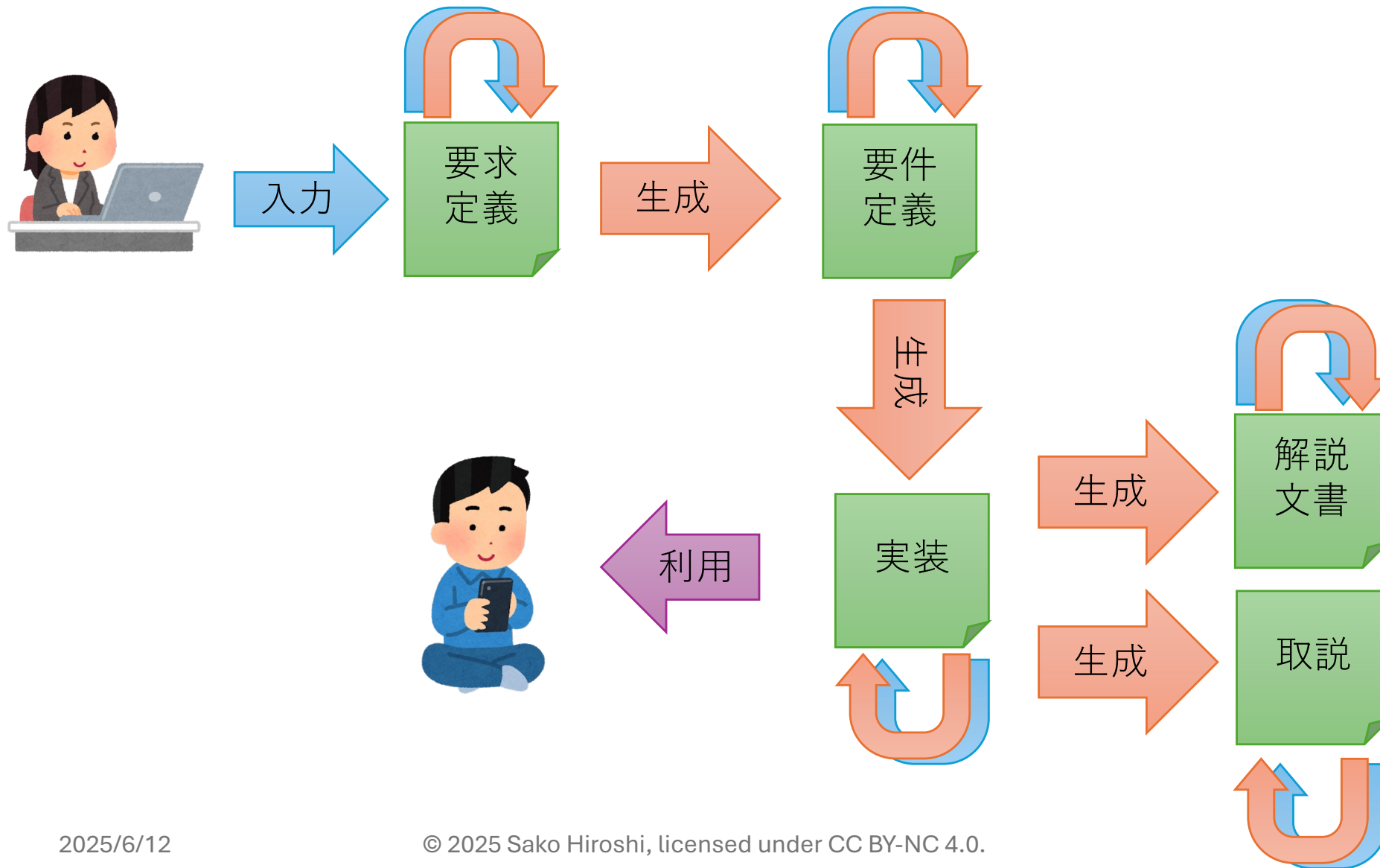
初の真にインテリジェントな開発環境として、Windsurf AI は強力な AI エージェントと直感的なコパイロットを組み合わせ、あなたのニーズに適應し、生産性を向上させる比類のないコーディング体験を生み出します。

[Windsurf AI IDE を体験する](#)

Windsurf  
Getting started with Windsurf  
Code with Cascade  
Open Command Palette  
Workspaces  
Open Folder  
Connect to...  
Generate a project with Cascade  
Write with Cascade  
Ask anything - use '@' to mention code blocks  
Claude 3.5 Sonnet

# A<sup>2</sup>P<sup>2</sup> の概要

今回の例題は A2P2 の中心ツールとして、Codeium の **Windsurf** を利用することにします。別のチャット型 AI も補助的に利用します



# 例題：買い物リスト

以下のプロンプトを ChatGPT に投入して、作りたいアプリケーションの要求定義の叩き台を作成する

「買い物リスト」を作成するアプリケーションを作りたいと思っています。

ウェブアプリとして構成し、商品の登録や、購入時にチェックマークを入れることができるようなものとします。情報は何らかの形で保存し、あとから参照できるようにしたいと思います。

このアプリケーションの「要求定義」をマークダウン形式で作成してください。

# 買い物リスト（要求定義）

## 買い物リストアプリケーション 要求定義

### 1. 概要

本アプリケーションは、ユーザが買い物リストを作成し、購入状況を管理するためのウェブアプリケーションである。ユーザはブラウザ上で商品を登録し、購入済みの商品にはチェックマークを入れることで管理できる。リストのデータは何らかの永続化ストレージに保存され、後日参照・編集ができるようにする。

### 2. 目的・背景

- 目的
  - ユーザが手軽に買い物リストを作成し、買い漏れを防ぐ。
  - 商品の購入状況を可視化し、必要なものを即座に把握できるようにする。
- 背景
  - 紙のメモや口頭の伝達では、買い漏れや二重購入が発生しやすい。
  - スマホやPCから簡単にアクセスできる買い物リストがあれば、日常的な買い物を効率化できる。



あくまでもアプリの機能要求が中心で、実装の詳細はまだ出てこない

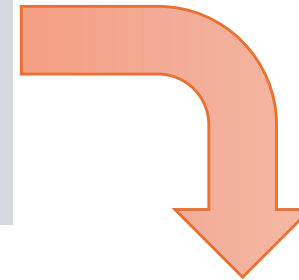
続きは →

<https://chatgpt.com/share/67cd3614-6f94-800b-9518-35f0579572e8>

Me

requirements.md の内容に従って システム要件定義を行い specification.md に記入してください。

なお 実装言語はクライアント側は HTML, CSS, JavaScript、サーバー側は Python とします。データベースとして sqlite を使うことにしてデータモデルも示してください。



## 要求定義から 要件定義へ

要件定義に基づいて、仕様書を以下のように更新しました：

1. システム概要を要件定義に合わせて具体化し、ブラウザベースのアプリケーションであることを明確にしました。
2. 機能要件を以下の6つの主要機能に整理しました：
  - 商品登録機能
  - 商品一覧表示機能
  - 購入チェック機能
  - 商品編集機能
  - 商品削除機能
  - データ永続化機能
3. データモデルを簡素化し、単一のItemsテーブルに統合しました。これは要件定義で示された「シングルユーザまたは複数ユーザでの共同利用」に対応できる柔軟な構造です。
4. 非機能要件を追加し、以下の項目を詳細化しました：
  - パフォーマンス（1秒以内の画面表示）
  - 信頼性（バックアップ、エラー処理）
  - ユーザビリティ（レスポンスデザイン）
  - 保守性（MVCアーキテクチャ）
5. 画面設計を追加し、ユーザインターフェースの詳細を記述しました。
6. テスト要件を追加し、単体テストとUI/UXテストの内容を明確化しました。

この仕様書は、要件定義で示された「シンプルで使いやすい買い物リスト管理アプリケーション」の実現に必要な技術的な詳細を提供しています。

# システム要件定義

## 買い物リストアプリケーション仕様書

### 1. システム概要

ユーザが買い物リストを作成し、購入状況を管理するためのウェブアプリケーション。ブラウザ上で商品を登録し、購入済みの商品にはチェックマークを入れることで管理できる。

### 2. 技術スタック

#### フロントエンド

- HTML5
- CSS3
- JavaScript (Vanilla JS)
- レスポンシブデザイン対応

#### バックエンド

- Python (Flask)
- SQLite3

### 3. 機能要件

#### 3.1 商品登録機能

- 商品名の入力フィールド (必須)
- 登録ボタン
- 入力バリデーション (空文字チェック)



続きは →

<https://github.com/ardbeg1958/ShoppingList01/blob/main/specification.md>

# 機能要件はできたがUIが心配

なので、UIがどのようなものになるのかを、モック画面を作成して確認

Me

ありがとうございます。では requirements.md, specification.md の内容に従って、画面のプロトタイプをまず作成し、人間にレビューさせてください。

HTMLとCSSを使用して、プロトタイプを作成します。まず必要なファイルを作成していきます。

## 画面のデザインを確認したい

Me ありがとうございます。では requirements.md, specification.md の内容に従って、画面のプロトタイプをまず作成し、人間にレビューさせてください。

HTMLとCSSを使用して、プロトタイプを作成します。まず必要なファイルを作成していきます。

プロトタイプが表示されました。このプロトタイプは要件定義と仕様書に基づいて、以下の機能を実装しています：

1. UI/UX
  - シンプルで直感的なデザイン
  - レスポンシブ対応（スマートフォン・タブレット表示に最適化）
  - 視認性の高いフォントとカラースキーム
  - 適切な余白とシャドウによる階層構造の表現
2. 基本機能
  - 商品の追加（入力フォーム）
  - 商品一覧の表示
  - 購入状態の切り替え（チェックボックス）
  - 商品名の編集（モーダルウィンドウ）
  - 商品の削除（確認ダイアログ付き）
3. インタラクション
  - 購入済み商品の視覚的表示（取り消し線と色変更）
  - 編集モーダルの表示/非表示
  - 削除時の確認ダイアログ
  - フォームバリデーション（空の商品名は登録不可）

プロトタイプをご確認いただき、以下の点についてフィードバックをいただけますと幸いです：

1. UIの使いやすさ
2. 視覚的なデザイン
3. 必要な機能の過不足
4. モバイル表示での使い勝手

修正が必要な点がございましたら、お申し付けください。

### 買い物リスト

商品名を入力してください

追加

牛乳

編集

削除

パン

編集

削除

卵

編集

削除

もずく

編集

削除

# 最終更新日時を表示するように依頼

Me リストの最終更新日時を表示するようにしてください。

リストの最終更新日時を表示するために、HTMLとCSSを修正し、JavaScriptで更新日時の管理を追加します。

まず、HTMLを更新して最終更新日時を表示する要素を追加します：

Edited `index.html` +3 -0 72 [Open diff](#)

CSSを更新して、最終更新日時の表示スタイルを追加します：

Edited `style.css` +8 -1 256 [Open diff](#)

JavaScriptを更新して、最終更新日時の管理機能を追加します：

Edited `main.js` +23 -0 125 [Open diff](#)

⚠️ 4 new lint errors [Auto-fix on](#) ▾

JavaScriptファイルにシンタックスエラーがあるようなので、修正します。

Edited `main.js` +0 -1 147 [Open diff](#)

No credits consumed by purely lint-fixing edit [?](#)

最終更新日時の表示機能を追加しました。以下の変更を行いました：

1. ヘッダー部分に最終更新日時を表示する要素を追加
2. 最終更新日時のスタイルを追加（中央寄せ、グレー色で控えめな表示）
3. 以下のアクションが行われた際に最終更新日時が自動的に更新されるように実装：
  - 新規商品の追加
  - 商品名の編集
  - 商品の削除
  - 購入状態の切り替え（チェックボックスの変更）

最終更新日時は「YYYY-MM-DD HH:mm」形式で表示され、日本語のロケール設定を使用しています。

ブラウザをリフレッシュすると、変更を確認できます。商品の追加、編集、削除、チェックボックスの切り替えなど、リストに変更を加えるたびに最終更新日時が自動的に更新されます。

# 修正結果

## 買い物リスト

最終更新: 2025/03/09 16:15

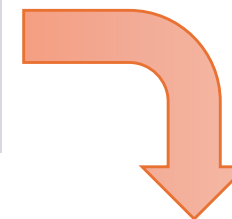
 追加

<input type="checkbox"/> 牛乳	<button>編集</button> <button>削除</button>
<input checked="" type="checkbox"/> パン	<button>編集</button> <button>削除</button>
<input checked="" type="checkbox"/> 卵	<button>編集</button> <button>削除</button>
<input type="checkbox"/> もずく酢	<button>編集</button> <button>削除</button>

# システム要件仕様に反映

Me ではこの変更を specification.md にも反映してください。

specification.mdに最終更新日時の表示機能を追加します。



```
24 ### 3.2 商品一覧表示機能
```

- 25 - 登録商品の一覧表示
- 26 - スクロール対応
- 27 - オプション：並び替え機能

```
28
```

```
29 ### 3.3 購入チェック機能
```

- 30 - 各商品に対するチェックボックス
- 31 - チェック済み商品のスタイル変更（取り消し線・背景色）
- 32 - チェック状態の切り替え機能

```
34 ### 3.4 商品編集機能
```

- 35 - インライン編集機能
- 36 - 編集内容の確定・キャンセル

```
38 ### 3.5 商品削除機能
```

- 39 - 削除ボタン
- 40 - 確認ダイアログ
- 41 - オプション：ゴミ箱/アーカイブ機能

```
43 ### 3.6 データ永続化機能
```

- 44 - SQLiteデータベースでのデータ保存
- 45 - REST APIによるCRUD操作

```
46
```

```
47 ## 4. データモデル
```

```
48
```

```
24 ### 3.2 商品一覧表示機能
```

- 25 - 登録商品の一覧表示
- 26 - スクロール対応
- 27 - オプション：並び替え機能

```
28+ - リストの最終更新日時の表示 (YYYY-MM-DD HH:mm形式)
```

```
29
```

```
30 ### 3.3 購入チェック機能
```

- 31 - 各商品に対するチェックボックス
- 32 - チェック済み商品のスタイル変更（取り消し線・背景色）
- 33 - チェック状態の切り替え機能

```
35 ### 3.4 商品編集機能
```

- 36 - インライン編集機能
- 37 - 編集内容の確定・キャンセル

```
39 ### 3.5 商品削除機能
```

- 40 - 削除ボタン
- 41 - 確認ダイアログ
- 42 - オプション：ゴミ箱/アーカイブ機能

```
44 ### 3.6 データ永続化機能
```

- 45 - SQLiteデータベースでのデータ保存
- 46 - REST APIによるCRUD操作

```
47+ - 全ての変更操作時に最終更新日時を自動更新
```

```
48
```

```
49 ## 4. データモデル
```

```
50
```

# コード改善

Me はい、改善手順を教えてください

改善手順を段階的に説明させていただきます。優先度の高い順に以下の3つのフェーズで進めていきましょう：

フェーズ1: データの整合性とエラーハンドリング（バックエンド）

1. データベース操作の改善
  - `query_db` 関数の改善（commit制御の追加）
  - トランザクション管理の強化
  - タイムスタンプ処理の統一
2. エラーハンドリングの強化
  - アイテム存在確認の追加
  - 入力値のバリデーション強化
  - エラーメッセージの統一

フェーズ2: フロントエンドの基本的な改善

1. エラーハンドリングとユーザーフィードバック
  - エラーメッセージ表示用のコンポーネント追加
  - 処理中の状態表示（ローディング）
  - 操作結果の通知
2. UIの基本的な改善
  - フォームのバリデーション強化
  - モーダルの挙動改善
  - ボタンの状態管理

フェーズ3: UIとUXの最適化

1. アクセシビリティの向上
  - ARIA属性の追加
  - キーボード操作のサポート
  - スクリーンリーダー対応
2. レスポンシブデザインの強化
  - モバイル表示の最適化
  - タッチ操作の改善

それでは、フェーズ1から始めていきましょうか？

Suggested Actions

はい、フェーズ1から始めましょう

いいえ、別のアプローチを考えましょう

# しかし実行できない ...

<http://localhost:5000/> にアクセスしてテストせよと言われるが、エラーが起きる。何度 `windsurf` に修正させても上手く行かない ... そこで `perplexity.ai` に聞いてみることにした。

`flask` アプリを実行しようとしているのですが、`python app.py` で実行しても

===

`localhost` へのアクセスが拒否されました  
このページを表示する権限がありません。

**HTTP ERROR 403**

===

と言われます。 `macOS` です。最も考えられる原因は何でしょう？

# ポートの競合と判明

flask アプリを実行しようとしているのですが、`python app.py` で実行しても

```
===  
loc  
この  
HT  
===  
と言
```

## ✖ 回答

最も考えられる原因は、macOS ではポート 5000 が AirPlay Receiver によって使用されているため、Flask アプリケーションのデフォルト設定と競合していることです。この問題は macOS Monterey 以降で特に顕著です。

## 解決方法

以下のいずれかの方法で問題を解決できます：

### 1. Flask のポートを変更する

Flask アプリケーションが使用するポートを 5000 以外に変更します。例えば、以下のようにコードを修正します：

```
python  
  
app.run(port=8000)
```



# 微調整とテストケースの提示

それでは、先ほど計画したテストケースを実行してみましょう。http://localhost:8000 にアクセスして、以下のテストを順番に実行してください：

## 1. 空の商品名のテスト

- 入力欄を空のまま「追加」ボタンをクリックしてください
- エラーメッセージ「商品名は必須です」が表示されるはず

## 2. 長すぎる商品名のテスト

- 以下の商品名を入力してください：

Copy Insert

とても長い商品名テストとても長い商品名テストとても長い商品名テストとても長い商品名テストとても長い商品名テストとても長い商品名テストとても長い商品名テスト

- エラーメッセージ「商品名は100文字以内で入力してください」が表示されるはず

## 3. 特殊文字を含む商品名のテスト

- 以下の商品名を入力してください：

Copy Insert

牛乳!!!@@@

- エラーメッセージ「商品名に使用できない文字が含まれています」が表示されるはず

## 4. 正常な商品名のテスト

- 以下の商品名を順番に入力してください：

Copy Insert

牛乳  
食パン  
卵

- 各アイテムが正常に追加され、ローディングスピナーが表示されるはず
- 最終更新時刻が更新されるはず

テストを実行してみて、結果を教えてください。

# 最終画面

## 買い物リスト

最終更新: 2025/03/09 17:22

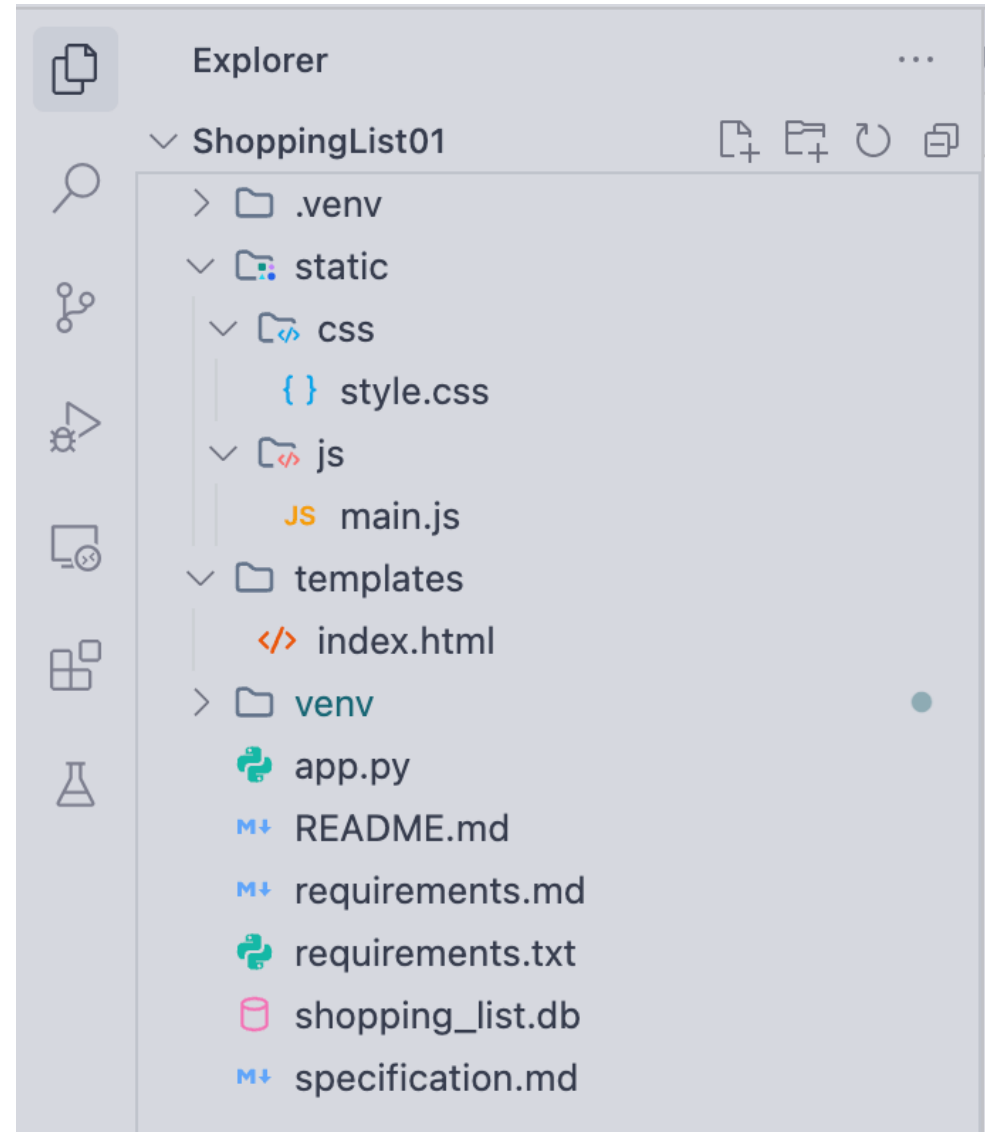
<input type="checkbox"/> 牛乳	<input type="button" value="編集"/>	<input type="button" value="削除"/>
<input type="checkbox"/> 食パン	<input type="button" value="編集"/>	<input type="button" value="削除"/>
<input type="checkbox"/> 卵	<input type="button" value="編集"/>	<input type="button" value="削除"/>

# ディレクトリ構造

最初の requirements.md  
以外は windsurf が生成し  
た



GitHub



# マニュアルの生成

Me

requirements.md, specification.md, その他のコードを参照して、操作マニュアル manual.md を生成してください。



## 買い物リストアプリケーション 操作マニュアル

### 1. はじめに

このアプリケーションは、日常の買い物をより効率的に管理するためのWebアプリケーションです。商品の登録から購入状況の管理まで、シンプルな操作で買い物リストを管理できます。

### 2. アプリケーションの起動方法

1. ブラウザ（Chrome, Firefox, Safari等）を起動します
2. アドレスバーに `http://localhost:8000` を入力してアクセスします

### 3. 基本操作

#### 3.1 商品の登録方法

1. 画面上部の新規商品追加ボタンをクリックします
2. 商品名入力フィールドに商品名を入力します
  - 入力可能な文字：
    - 日本語（ひらがな、カタカナ、漢字）
    - 英数字
    - スペース、カンマ(,), ピリオド(.), ハイフン(-)
  - 制限事項：
    - 空白は登録できません
    - 最大100文字まで入力可能です
3. 登録ボタンをクリックして商品を追加します

# プログラム解説書の生成

Me Flask ライブラリ初心者のために、プログラムの内容を説明する program\_guide.md を生成してください。python, javascript などの言語の知識はあるものとします。

## 買い物リストアプリケーション プログラムガイド

### 1. プログラム構成概要

このアプリケーションは、Flask フレームワークと SQLite データベースを使用した RESTful API サーバーです。主な技術スタックは以下の通りです：

- バックエンド: Flask (Python)
- データベース: SQLite3
- フロントエンド: HTML/CSS/JavaScript

### 2. Flask アプリケーションの基本構造

#### 2.1 アプリケーションの初期化

```
from flask import Flask
app = Flask(__name__)
```

- `Flask` クラスのインスタンスを作成し、これがアプリケーションの中心となります
- `__name__` は現在の Python モジュール名を表す特殊変数です

#### 2.2 ルーティング

Flask ではデコレータを使用して URL ルートを定義します：

```
@app.route('/api/items', methods=['GET'])
def get_items():
    # 処理内容
```

途中で知識として与えた macOS での注意点が盛り込まれている

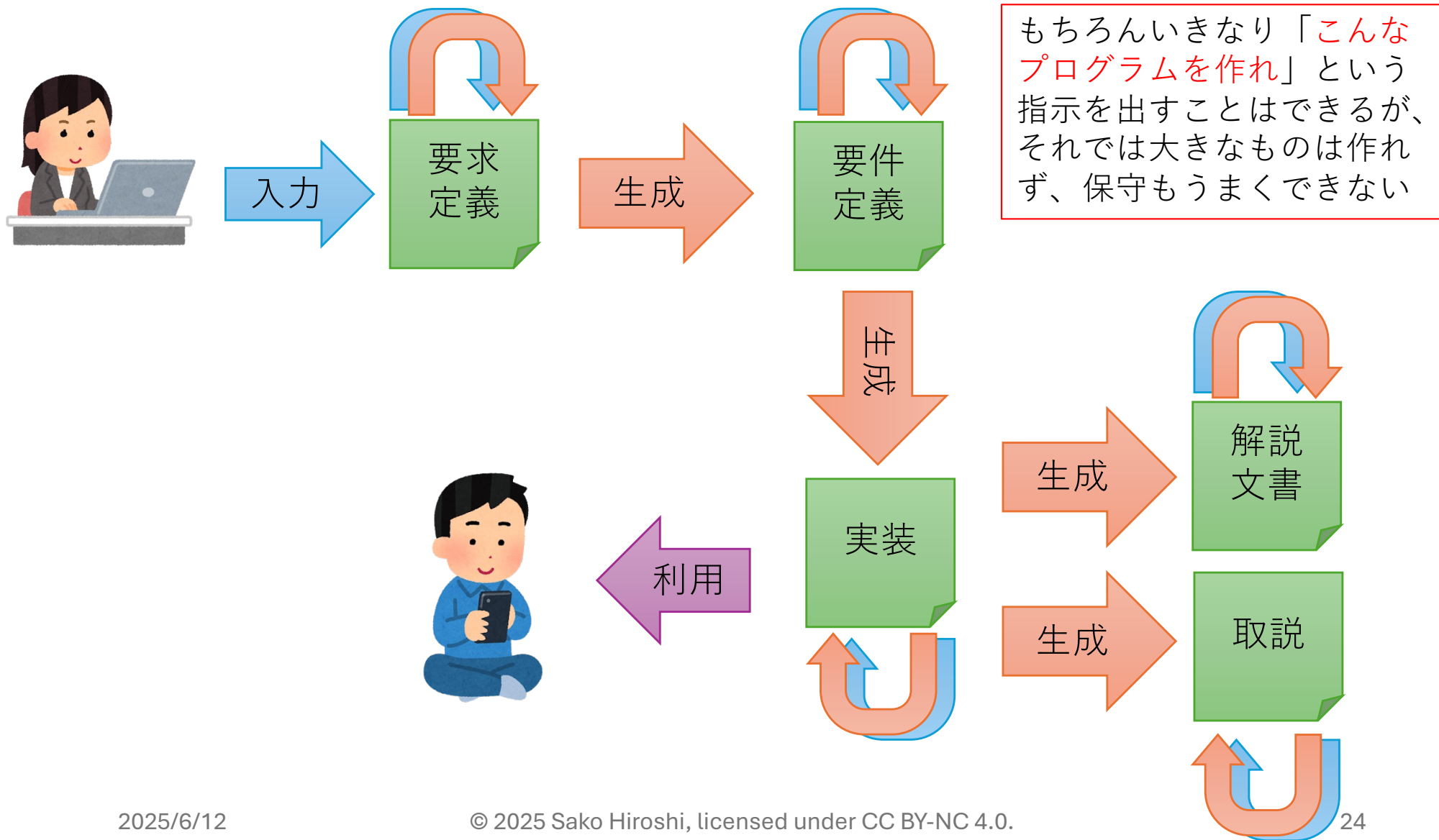
## 7. アプリケーションの起動

```
if __name__ == '__main__':  
    db_path = init_db() # データベースの初期化  
    app.run(debug=True, port=8000) # 開発サーバーの起動
```

- macOSではAirPlayとの競合を避けるため、ポート8000を使用します
- デバッグモードが有効な状態で起動します

# A<sup>2</sup>P<sup>2</sup> の特徴

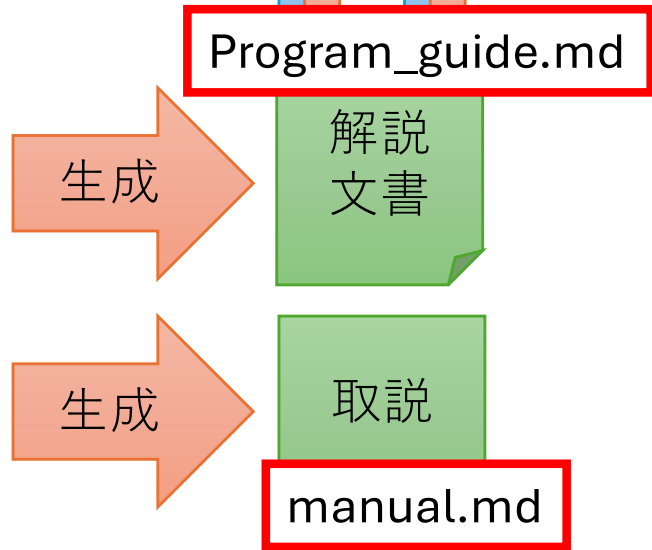
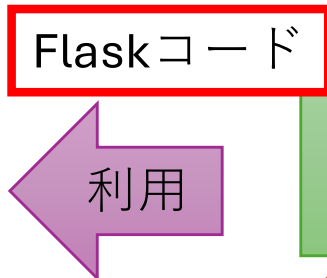
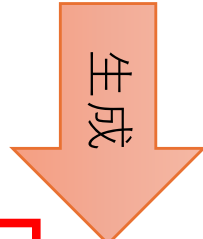
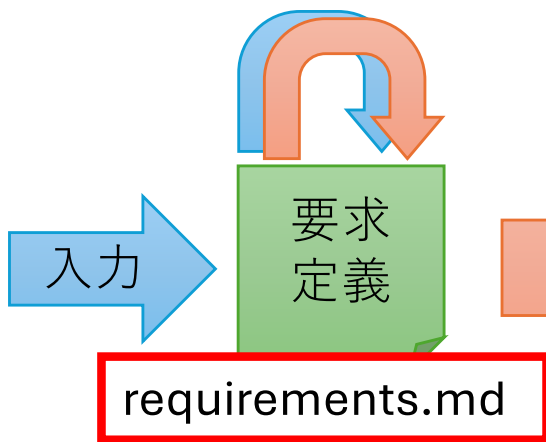
丸投げ型の AI 開発ツールとは違い、段階的に抽象度を下げていくことで、ある程度の規模のシステムや、実装方法の選択、保守などを行いやすくなる。これまでのやり方とも馴染む



# A<sup>2</sup>P<sup>2</sup> の特徴

今回のストーリーで生成された成果物

specification.md などから別途テストケースは生成可能

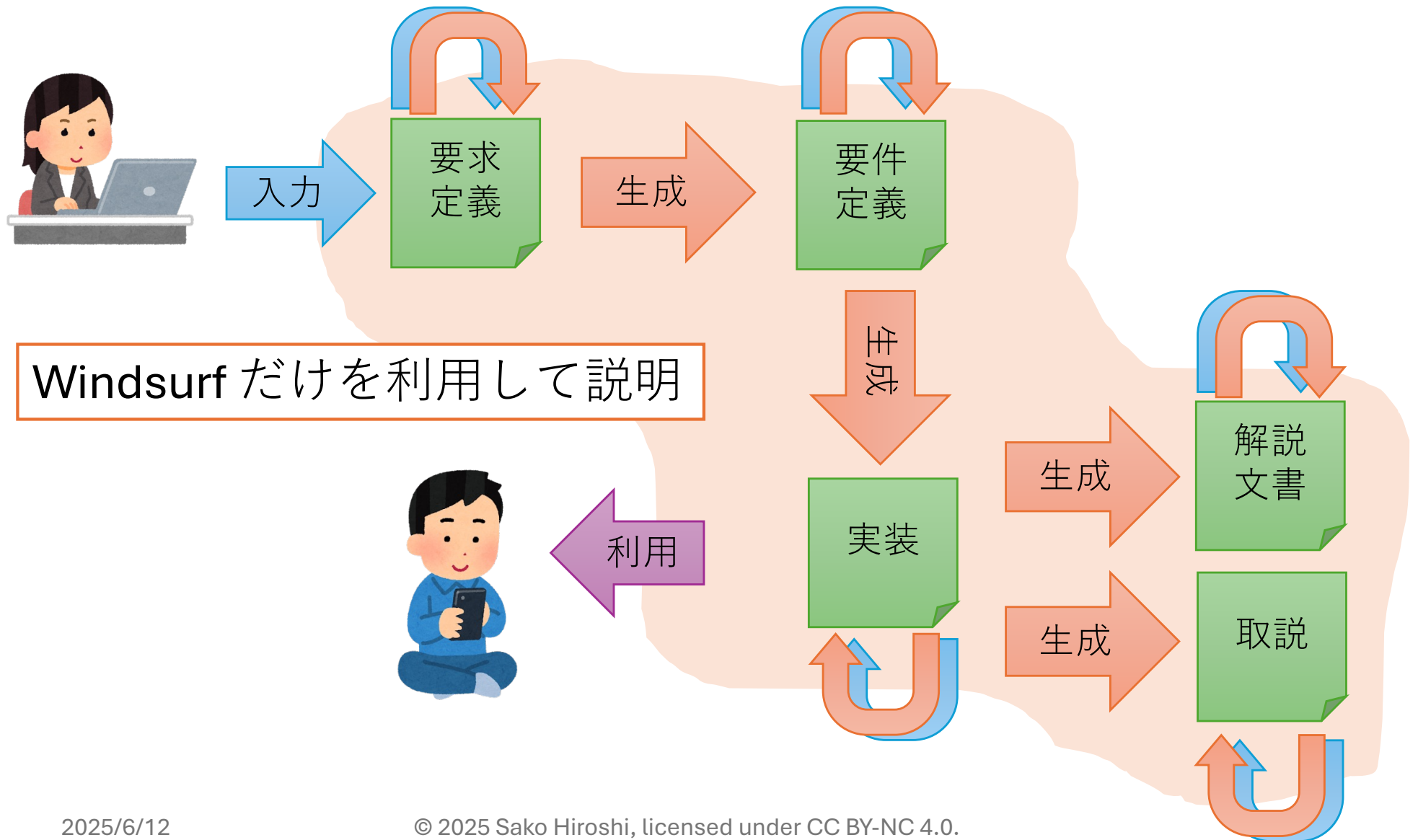


# A<sup>2</sup>P<sup>2</sup> の特徴

- 単なるチャットアプリとの違い
  - コードベースを理解してくれるので、セッションが中断しても再開可能
  - 同様に、既存の資産を読み込ませることができるので保守作業にも利用できる可能性
- **Bolt.new, Replit Agent** などのノーコード系 AI ツールとの違い
  - ノーコード系ツールは便利だが、細かい調整が難しく保守をしようとしても部分的に修正することは難しい（そもそもコードを人間が修正することをあまり考えていない）
  - **Windsurf, Cursor** などの A<sup>2</sup>P<sup>2</sup> ツールは自分のワークフローの中に柔軟に組み込むことができる

# ツールチェーン考

# プレゼンで説明したツールチェーン



# 最新ツールチェーン（案）

- 要求定義（ChatGPT / Claude / NotebookLM 等の Chat UI）
  - 対話でニーズや課題を深掘り
  - 抜け漏れや矛盾のない要件整理
- 要件定義・プロトタイプ（Windsurf / Cursor などの AI搭載エディタ）
  - 要求を明確な仕様へ落とし込む
  - 解説・背景付きで高品質な設計書生成
- 実装・保守（Codex/Jules/Claude Codeなどのエージェント型開発環境）
  - 仕様に基づき効率的にコード生成
  - テスト・保守も自動化

# 最新ツールチェーン (案)

