

生成AIの ソフトウェア開発への応用

AI in the Loop

酒匂寛

Designers' Den Corp

自己紹介

酒匂寛

Designers' Den Corp.

自己紹介 1/3

1980年代

ツールの時代

- ワークステーション、ソフトウェアツール、CASEの開発
 - 開発したのは統合プロダクトライン開発支援環境
 - 一年強で35万ステップのCプログラムを開発し、しばし廃人状態に
- B. Meyer の「オブジェクト指向入門」（アスキー）を翻訳し、オブジェクト指向とソフトウェア工学に目覚める
 - 契約による設計（DbC）、開放閉鎖原則

自己紹介 2/3

1990年代

00の時代

- オブジェクト指向に胸までつかる
 - 大規模プロジェクトの混乱が、仕様の不徹底と曖昧さ、非定量的なプロジェクト管理に起因することを思い知る
- 大規模分散オブジェクト指向プロジェクト
 - 医療系、物流系（どちらも～3000人月）
- 様々な組込機器ソフトのリエンジニアリング（構造整備）
 - ATM、自動精算機、POS端末、自動改札、交通管制、プリンタ、スキャナ、携帯電話などの既存の実装、設計を見直して構造化（フレームワーク化）
- 80年代の経験からドメイン指向開発環境の重要性を痛感
 - M.Jackson の「ソフトウェア要求と仕様」を訳し、その思いを更に深める
 - 運賃精算機開発のための専用記述言語と検証環境

自己紹介 3/3

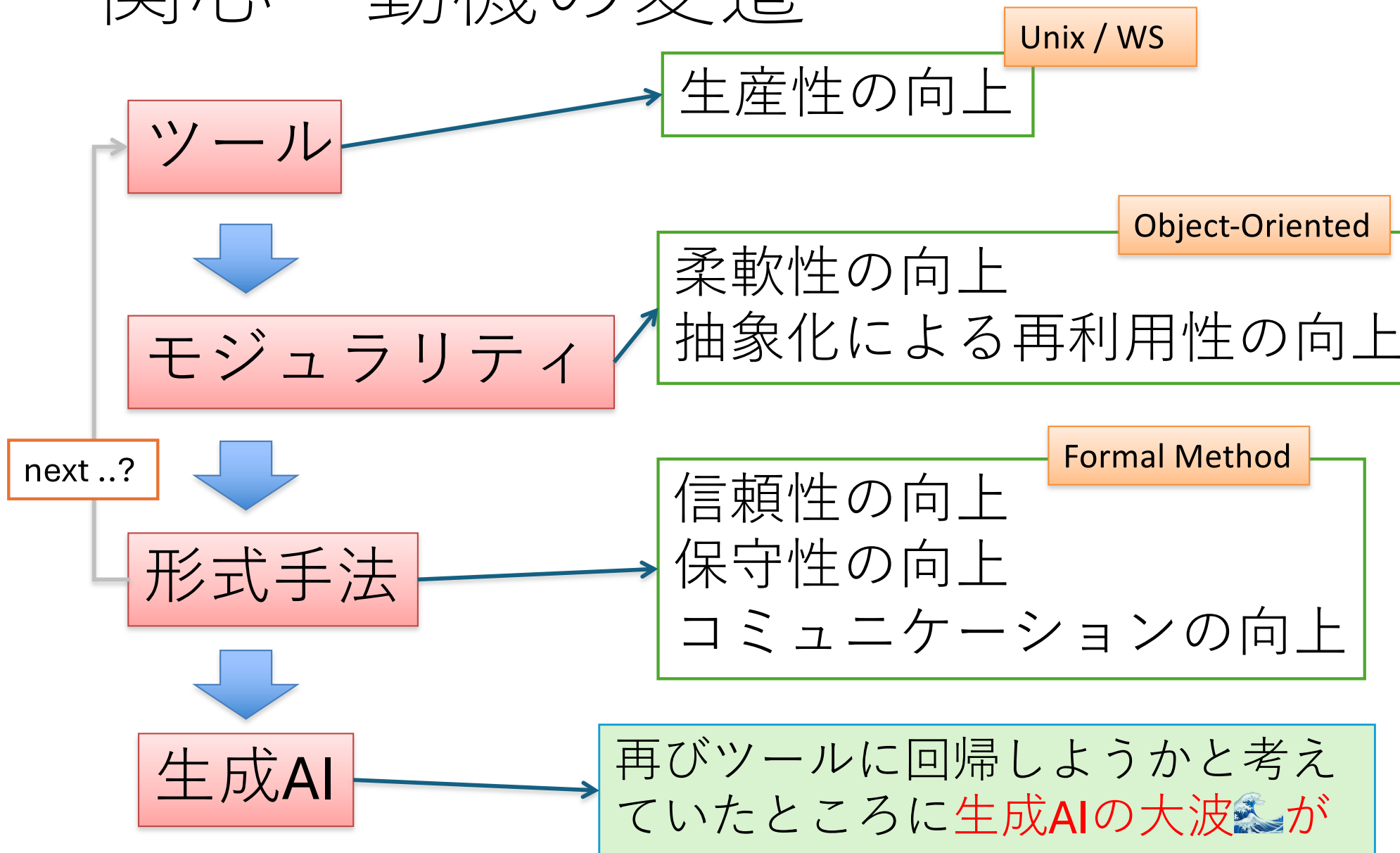
2000年代

形式手法の時代

- 問題領域を直接構成する手法に興味（仕様記述 → 実装解の導出）
 - 技術／工程を「対象問題領域」に沿って最適化する過程
 - ex. プリンタエンジンのプロダクトライン化
 - フレームワークと開発（分析、設計、実装、資産化）ガイドライン
- アーキテクチャ、フレームワーク中心のソフトウェア構成法に興味を覚える
 - ex. Gofu (good-old fashioned office)
 - 形式手法（VDM++, モデル検査等）をこれらの試みに組み込んで使いたいと思っている
 - ケアレスミス排除、仕様書品質の向上、検証品質の向上、よりよい文書化、V&V
- 形式仕様記述言語の利用
 - オンライン証券システムの仕様記述 → オプション、マル優システム
 - 非接触ICカードチップの仕様記述 → モバイルフェリカ（仕様バグ0）
 - FAプログラミング言語の意味定義 → 言語意味定義
 - 国内某チップメーカーの仕様記述 → 外部IF・ハードウェアIF

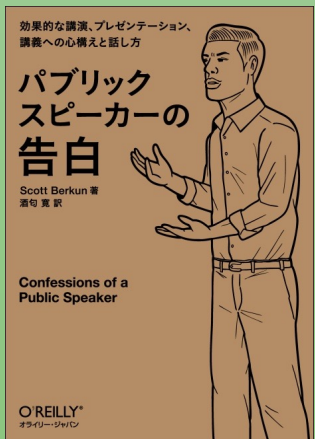
この頃一番 Ruby をヘビーに使っていた: ETLツール、動くフレームワーク仕様書、etc...

関心・動機の変遷

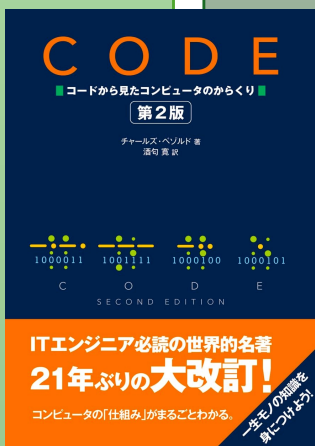


出版書籍

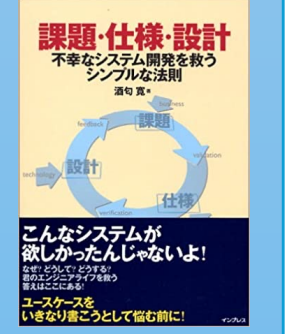
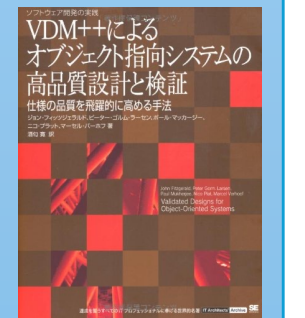
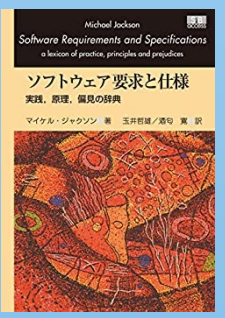
教養・一般



教育



専門



生成 AI 関連書籍



2024年9月13日発売

本日の講演の内容をカバーしています。

出版社サイト

<https://bookplus.nikkei.com/atcl/catalog/24/08/21/01557/>

話題

- 生成AIの現状
- 私たちが手に入れた道具
- 直接的な利用法
- ソフトウェア開発のライフサイクル
 - アイデアからBMCへ
 - BMCからTiDへ
 - 厳密な仕様の策定
 - 仕様から実装へ
- 組織内知識共有
- 「伝えること」と「翻訳する」こと
- プロンプトエンジニアリング (?)
- 考察とまとめ

生成AIの現状

生成AIの現状

- 大規模言語モデルの登場
 - 2022年末に OpenAI が ChatGPT を発表、文脈を意識した高度な対話を可能にした。無料で提供されていることもあり一気にブレイク
- テック企業各社はそれぞれの取り組みを行っている
 - OpenAI
 - 革新的な技術開発と商用化の両立
 - API提供とプロダクト開発の二軸展開
 - Google/DeepMind
 - 基礎研究から応用まで幅広い取り組み
 - Googleのインフラとの統合
 - Anthropic
 - 高いコーディング能力や文章作成能力
 - 様々なモデルを取り込んで組み合わせる柔軟性

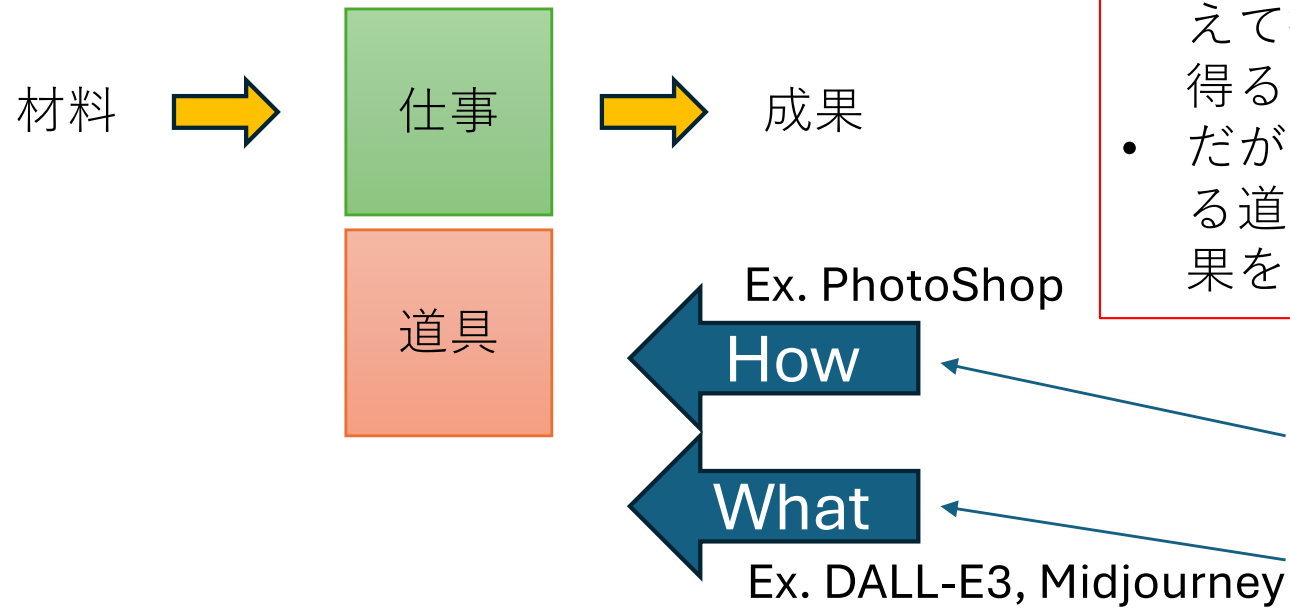
AI 検索の台頭

- **Perplexity**
 - AI 検索の先駆け、検索結果の根拠を示す
- **Genspark**
 - AI 検索に加えて Sparkpage というまとめを行ってくれる機能
- **Felo**
 - AI 検索の結果をマインドマップやプレゼンテーションにまとめてくれる機能
- 最近では **Deep Research** ブーム
 - AI ベンダー各社がこれまでとは比較にならない深い検索と要約を行う機能を出している

私たちが 手に入れた道具

Whatの道具 vs Howの道具

「道具」とは？



- これまでの道具は **How** を与えて微調整しながら成果を得るものだった
- だが生成 AI によって作られる道具は **What** を与えて成果を得るものとなる



Howの道具とWhatの道具

- How の道具（旧来の道具）
 - 人間が最終成果物に対して**詳細で明確なイメージ**を持つ（**具体的目標**）
 - 細かい指示・操作を人間が積み重ねて最終成果物を生み出す
 - **品質保証は人間の責任**
- What の道具（生成 AI が生み出す道具）
 - 人間は最終成果物に対する**方向性と性質**を指定する（**抽象的目標**）
 - 詳細は機械が生成し最終成果物とする
 - **品質保証は人間の責任**

Howの道具とWhatの道具

	利点	欠点
Howの道具	満足行くまで品質を追求することができる (?)	使いこなしの訓練に時間がかかる
Whatの道具	使いこなしの訓練にはあまり時間がかからない	自分の手を動かさないために品質に関する要求が低くなる (?)

どちらも最終成果物に関する品質保証が求められることは同じ。Whatの道具の方が最終成果物の性質（仕様）に関する検討に時間をかけられることになる？

直接的な利用法

ソフトウェア開発

コーディングアシスト (1/3)

ChatGPTにコード生成を依頼

【人間】 指定したURLにアクセスして、そのページのタイトルを出力する Python のコードを作成

----- (ここから下↓がGenerative AIの出力。以下同じ)

以下はPythonのrequestsとbeautifulsoup4ライブラリを使用して、指定したURLにアクセスし、そのページのタイトルを出力するコードです。

```
```python
import requests
from bs4 import BeautifulSoup

url = "https://www.example.com"
response = requests.get(url)
soup = BeautifulSoup(response.content, "html.parser")
title = soup.title.string

print(title)
```
```

まず、requestsライブラリをインポートしてURLにアクセスします。そして、beautifulsoup4ライブラリを使ってHTMLページから情報を抽出します。ここで、response変数にHTMLページの内容が格納され、BeautifulSoupオブジェクトのsoupを作成して、そのタイトルを取得し、最後に出力します。

コードが生成されるだけでなく、その説明も同時に生成されている

この「サービス精神の旺盛さ」がChatGPTの特徴の一つ

コーディングアシスト (2/3)

URLが固定だったので、引数として与えられるように依頼

【人間】url をコマンドラインから取得できるようにして下さい

以下は、Python の requests ライブラリを使用して、指定された URL からページタイトルを取得し、出力するコード例です。また、sys ライブラリを使用して、コマンドラインから url を受け取ります。

```
```python
import sys
import requests
from bs4 import BeautifulSoup

コマンドラインから url を受け取る
url = sys.argv[1]

requests ライブラリを使用して、HTML ページを取得する
response = requests.get(url)

BeautifulSoup ライブラリを使用して、HTML ページからタイトルを抽出する
soup = BeautifulSoup(response.text, 'html.parser')
title = soup.find('title').text

タイトルを出力する
print(title)
```
```

引数の取り込みが追加された

このコードを `get_title.py` という名前で保存し、コマンドラインから次のように実行できます。

```
python get_title.py https://www.google.com/
```

コーディングアシスト (3/3)

URLがないときはエラーで終了するように依頼

【人間】 指定した URL が存在しないときにはエラーメッセージを出して終了するようにして下さい。

以下に、指定した URL からタイトルを取得する Python コードを示します。

```
```python
import sys
import requests
from bs4 import BeautifulSoup

コマンドライン引数から URL を取得する
url = sys.argv[1]

URL にアクセスして HTML を取得する
response = requests.get(url)

ステータスコードが 200 以外の場合はエラーメッセージを出力して終了する
if response.status_code != 200:
 print(f"Error: {response.status_code}")
 sys.exit()

BeautifulSoup を使って HTML からタイトルを取得する
soup = BeautifulSoup(response.content, 'html.parser')
title = soup.title.string

タイトルを出力する
print(title)
```
```

引数が足りない場合は考えていない

URLが存在しないときという指定を拡大解釈して対処している

このコードでは、まず `sys.argv[1]` を使ってコマンドラインから URL を取得し、`requests.get()` を使ってその URL にアクセスして HTML を取得します。次に、`response.status_code` をチェックして、ステータスコードが 200 でない場合にはエラーメッセージを出力してプログラムを終了します。最後に、`BeautifulSoup` を使って HTML からタイトルを取得し、それを出力します。

レビューアシスト

先のコードに問題がないかレビューを依頼、問題があれば指摘し修正案を出す

【人間】 Find problems with the following code

このコードにはいくつかの問題があります。エラー処理や例外処理が不足しています。以下は、修正したコードです。修正点についてはコメントで説明しています。

```
```python
import sys
import requests
from bs4 import BeautifulSoup

コマンドライン引数から URL を取得する
try:
 url = sys.argv[1]
except IndexError:
 print("Error: URL not specified")
 sys.exit()

URL にアクセスして HTML を取得する
try:
 response = requests.get(url)
except requests.exceptions.RequestException as e:
 print(f"Error: {e}")
 sys.exit()

ステータスコードが 200 以外の場合はエラーメッセージを出力して終了する
if response.status_code != 200:
 print(f"Error: {response.status_code}")
 sys.exit()
```
```

```
# BeautifulSoup を使って HTML からタイトルを取得する
soup = BeautifulSoup(response.content, 'html.parser')
title = soup.title.string
```

```
# タイトルを出力する
print(title)
```
```

変更内容：

- \* `url` の取得部分で `IndexError` が発生する可能性があるため、`try-except` 文で例外処理を追加しています。
- \* `requests.get()` の例外処理を追加しています。
- \* ステータスコードが200以外の場合はエラーメッセージを表示し、`sys.exit()` を使用してプログラムを終了するようにしました。

# ここまでの手順

- 仕事を依頼
  - ここではプログラムの生成を依頼
- 仕事の仕様変更を依頼
  - 引数の追加
- 仕事の仕様変更を依頼
  - 例外処理の追加
- 結果のレビューを依頼
  - 問題点を指摘しさらに処理を追加

生成

ChatGPTの優れている（面白い）点は、対話を重ねて答を生成していけること

内省

生成した結果を評価させることもできる

# 注意

- ここまで調子よく進んできたが ...
- ここで作られたプログラム（成果物）は  
正しいかどうかは保証されない
- なので最終成果物（中間成果物も）の正し  
しさは人間が確認しなければならない

これは常に気をつけておかなければならないこと

# 解析アシスト (1)

形式仕様記述言語VDM-SL のコードの解析を依頼

【人間】以下の VDM-SL のコードを説明して下さい。

**types**

金額型 = `int`;

商品券型 ::

商品券媒体 : 商品券媒体型

商品券額面 : 金額型;

商品券媒体型 = `<電子>|<紙>`;

商品券セット型 ::

商品券媒体 : 商品券媒体型

商品券構成 : `set of` 商品券型

**inv** ts ==

ts.商品券媒体 = `<電子>` => `forall t in set ts.商品券構成 & t.商品券媒体 = <電子>`  
`and`

ts.商品券媒体 = `<紙>` => `forall t in set ts.商品券構成 & t.商品券媒体 = <紙>`;

# 解析アシスト (2)

【人間】プログラミングに関する知識がない業務専門家にわかるように、説明して下さい。

このコードは、商品券に関する情報を整理するためのルールを定義しています。以下の4つの要素が含まれています。

- **金額:** これは、お金の金額を表します。
- **商品券:** これは、商品券に関する情報を表すための要素で、2つの部分から構成されています。
  - **商品券のタイプ:** 電子タイプの商品券か、紙のタイプの商品券かを示します。
  - **商品券の金額:** 商品券の価値（お金の金額）を示します。
- **商品券のタイプ:** これは、商品券が電子タイプか紙タイプかを示す要素です。
- **商品券のグループ:** これは、同じタイプの商品券をまとめたもので、2つの部分から構成されています。
  - **商品券のタイプ:** グループ内のすべての商品券のタイプを示します（電子か紙か）。
  - **商品券のリスト:** グループに含まれる商品券の一覧を示します。

このコードには、**商品券のグループに関する特別なルール**が定義されています。このルールは、商品券のグループ内のすべての商品券が同じタイプであることを保証します。例えば、商品券のグループが電子タイプの場合、そのグループ内のすべての商品券は電子タイプでなければならないということです。同様に、商品券のグループが紙タイプの場合、そのグループ内のすべての商品券は紙タイプでなければならないということです。

# 解析アシスト (3)

【人間】 この VDM-SL を形式仕様として受け取って実際のプログラミングを行う **エンジニア向け** の解説を行って下さい。

この VDM-SL 形式仕様は、商品券に関するデータ構造を定義しています。プログラマーとして、この仕様を実際のプログラムに変換する際に、以下のデータ構造と制約を考慮する必要があります。

- **金額型:** これは整数型で、金額を表します。プログラムで金額を扱う際に、整数型を使用してください。
- **商品券型:** これは構造体で、商品券の情報を表します。プログラムでは、クラスや構造体を使用して、以下の2つのプロパティを持つオブジェクトを定義します。
  - **商品券媒体:** 商品券が電子媒体か紙媒体かを表す列挙型を定義します。例えば、C++ではenum class、Javaではenum、PythonではEnumを使用します。
  - **商品券額面:** 金額型（整数型）を使用して、商品券の額面金額を表します。
- **商品券セット型:** これは構造体で、同じ媒体の商品券のセットを格納します。プログラムでは、クラスや構造体を使用して、以下の2つのプロパティを持つオブジェクトを定義します。
  - **商品券媒体:** 商品券媒体型を使用して、商品券セット内のすべての商品券の媒体を表します。
  - **商品券構成:** 商品券型のコレクション（リスト、セット、配列など）を使用して、商品券セットに含まれる商品券のリストを表します。

前のスライドでは「**商品券のグループに関する特別なルール**」と書かれていた

この仕様には、**商品券セット型に対する制約（不変条件）**が定義されています。この制約を遵守するために、商品券セットに商品券を追加する際には、追加しようとする商品券の媒体がセットの媒体と一致することをチェックするメソッドや関数を実装してください。例えば、クラスのメソッドやコンストラクタ内でチェックを行い、不一致があれば例外を投げるようにします。

実際にプログラムを実装する際は、使用するプログラミング言語の機能や慣習に従って、これらのデータ構造と制約をコードに反映させてください。

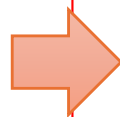
# 解析アシスト (4)

```
import sys
import requests
from bs4 import BeautifulSoup
from urllib.parse import urlparse

def is_valid_url(url):
 try:
 result = urlparse(url)
 return all([result.scheme, result.netloc])
 except ValueError:
 return False

def get_page_title(url):
 try:
 response = requests.get(url)
 if response.status_code == 200:
 soup = BeautifulSoup(response.content, "html.parser")
 title = soup.title.string if soup.title else "No title found"
 return title
 else:
 return f"Error: {response.status_code}"
 except requests.RequestException as e:
 return f"Error: {str(e)}"

if __name__ == "__main__":
 if len(sys.argv) > 1:
 url = sys.argv[1]
 if is_valid_url(url):
 title = get_page_title(url)
 print(title)
 else:
 print("Error: Invalid URL")
 else:
 print("Usage: python get_page_title.py <URL>")
```



This Python script is used to fetch and display the title of a web page given its URL. The script checks if the provided URL is valid, retrieves the HTML content of the page, and then extracts the title using the BeautifulSoup library. If the script is run without providing a URL or if the provided URL is invalid, it will display an error message.

Here's a step-by-step explanation of the code:

1. Import the necessary libraries: `sys`, `requests`, `BeautifulSoup`, and `urlparse`.
2. Define a function `is_valid_url(url)` to check whether a given URL is valid or not.
3. Define a function `get_page_title(url)` to fetch the web page title for a given URL.
4. Check if the script is being run as the main module. If it is, proceed to the next steps.
5. Check if a command-line argument (URL) is provided. If not, print the correct usage message.
6. If a URL is provided, check whether it's valid using `is_valid_url()`. If the URL is invalid, print an error message.
7. If the URL is valid, get the web page title using `get_page_title()` and print it.

Usage:

To use this script, run the following command in your terminal:

```
mathematica
```

Copy code

```
python get_page_title.py <URL>
```

Replace `<URL>` with the web page URL you want to fetch the title for. For example:

```
arduino
```

Copy code

```
python get_page_title.py https://www.example.com
```

The script will then output the title of the web page, or an error message if there's an issue with the URL or fetching the content.

話の前提  
生成**AI**と自動化

酒匂寛

Designers' Den Corp.

sakoh@ba2.so-net.ne.jp

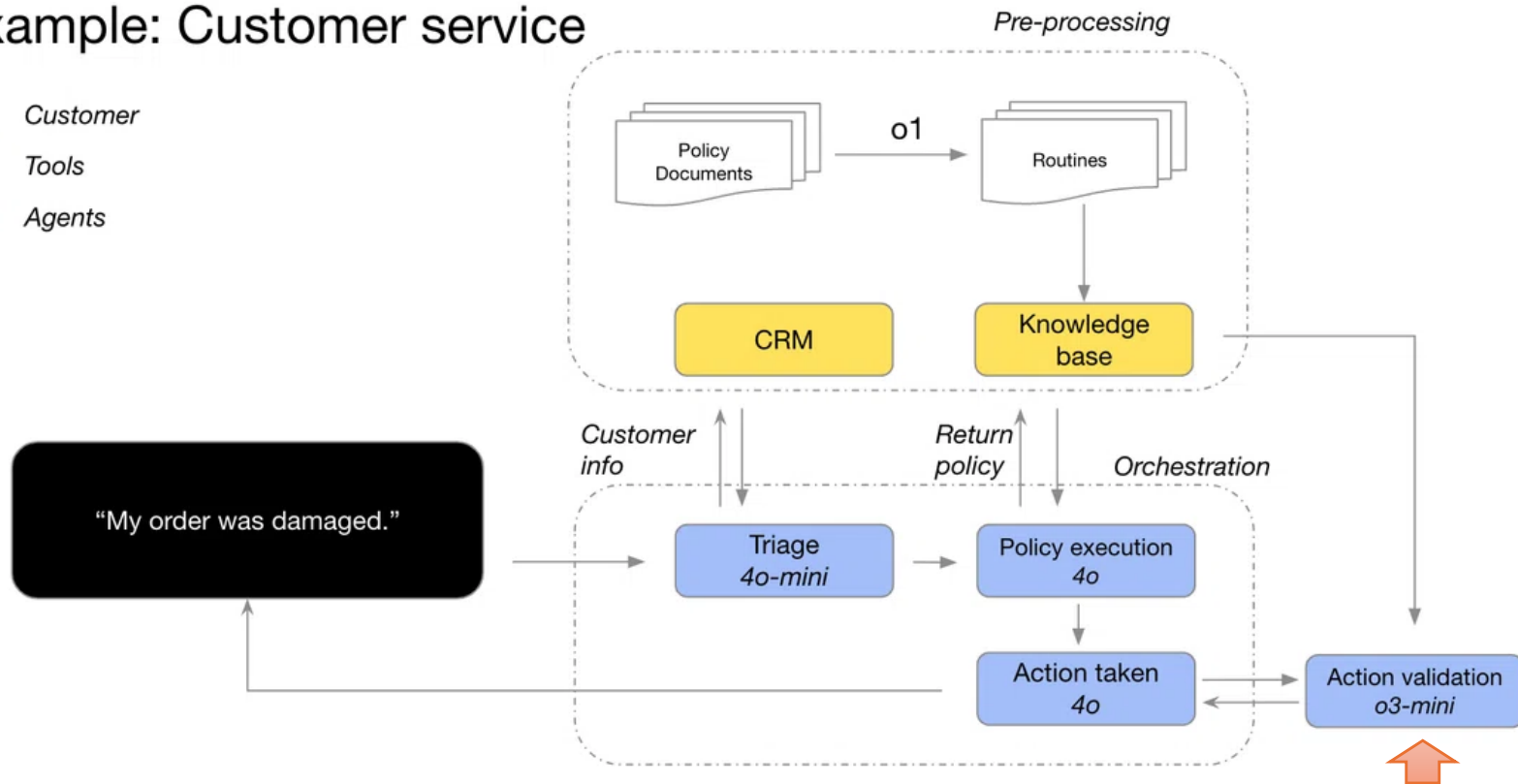


OpenAI のサイトには、AI をフルに使ったアプリケーションの例として以下のような図が出ている

CRM と手順書 (routine) を Orchestration Engine が使ってカスタマーサービスを実行するという図になっている

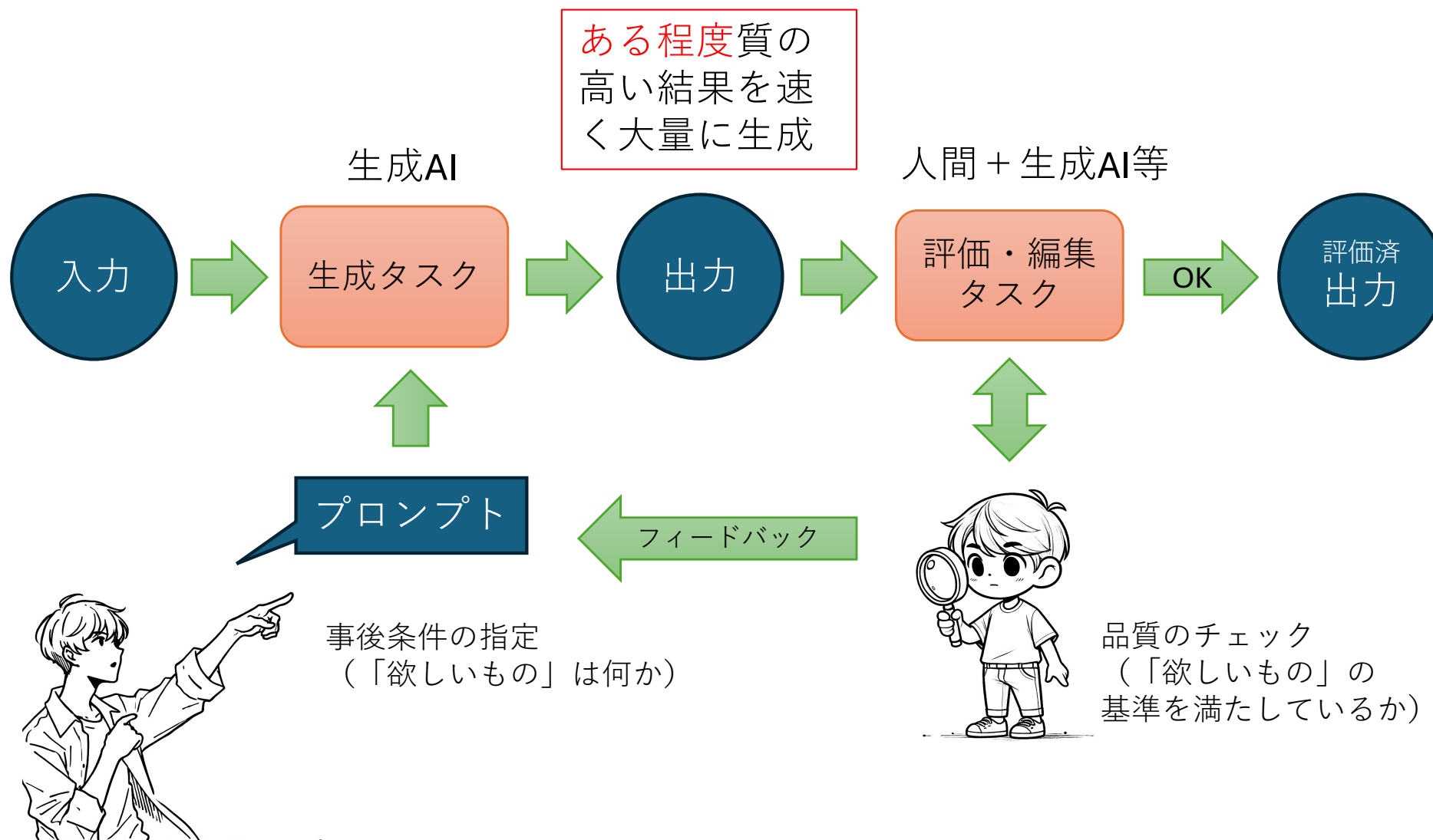
### Example: Customer service

- Customer
- Tools
- Agents

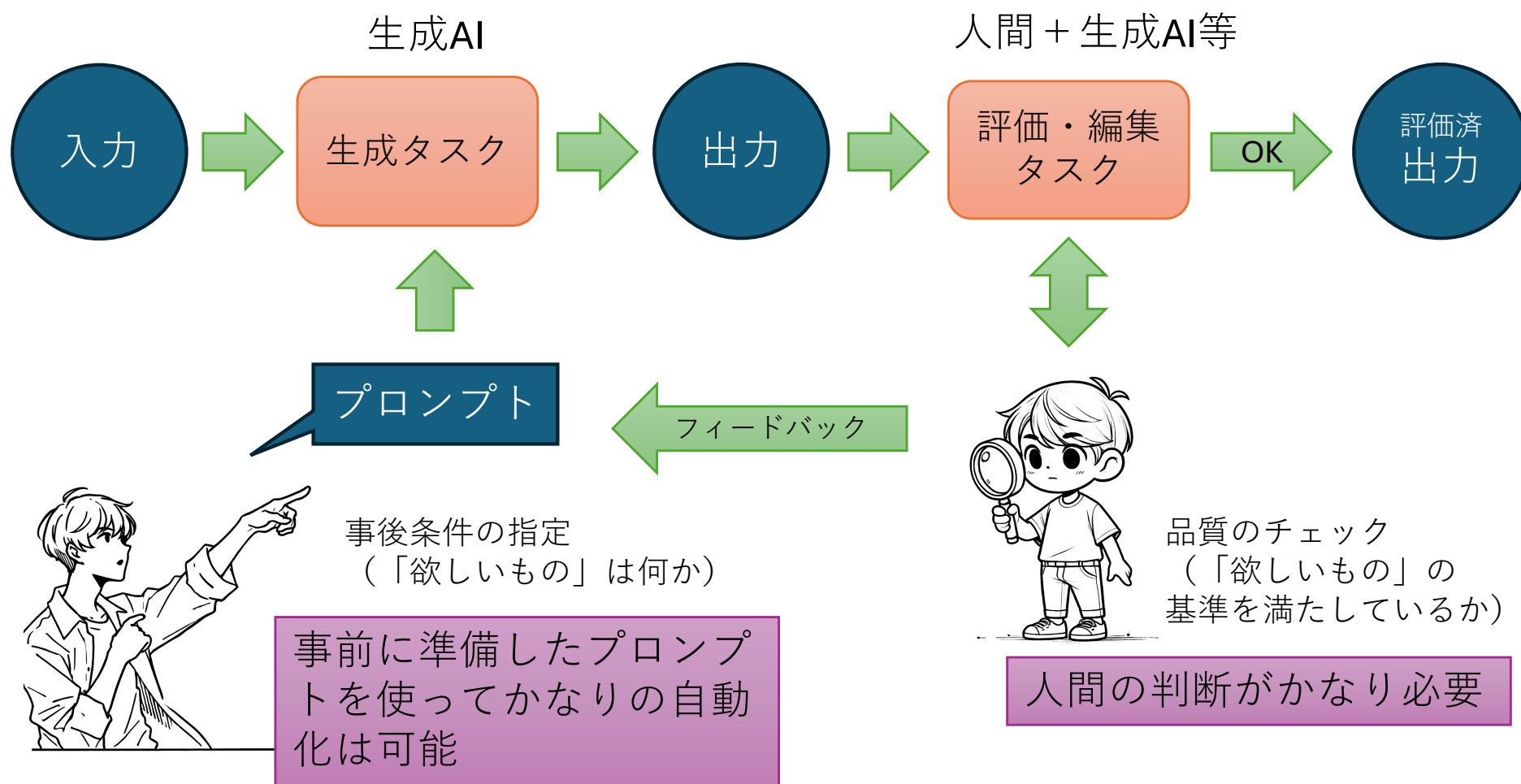


ここの出来が問題になる

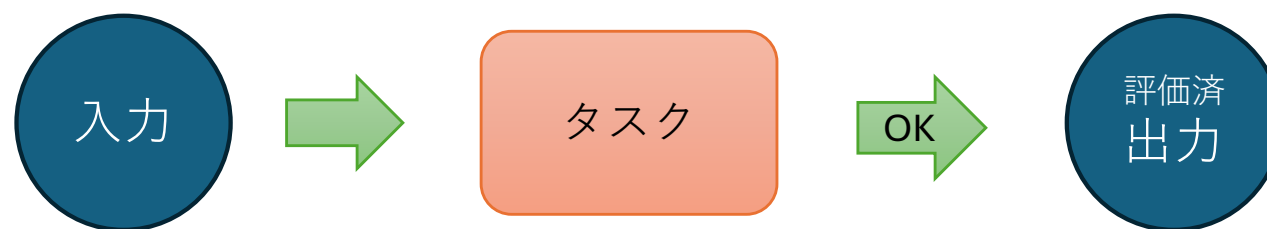
# 生成AIによるタスクの**実行**



# 生成AIによるタスク成果の検証



# 生成AIによる生産性向上？



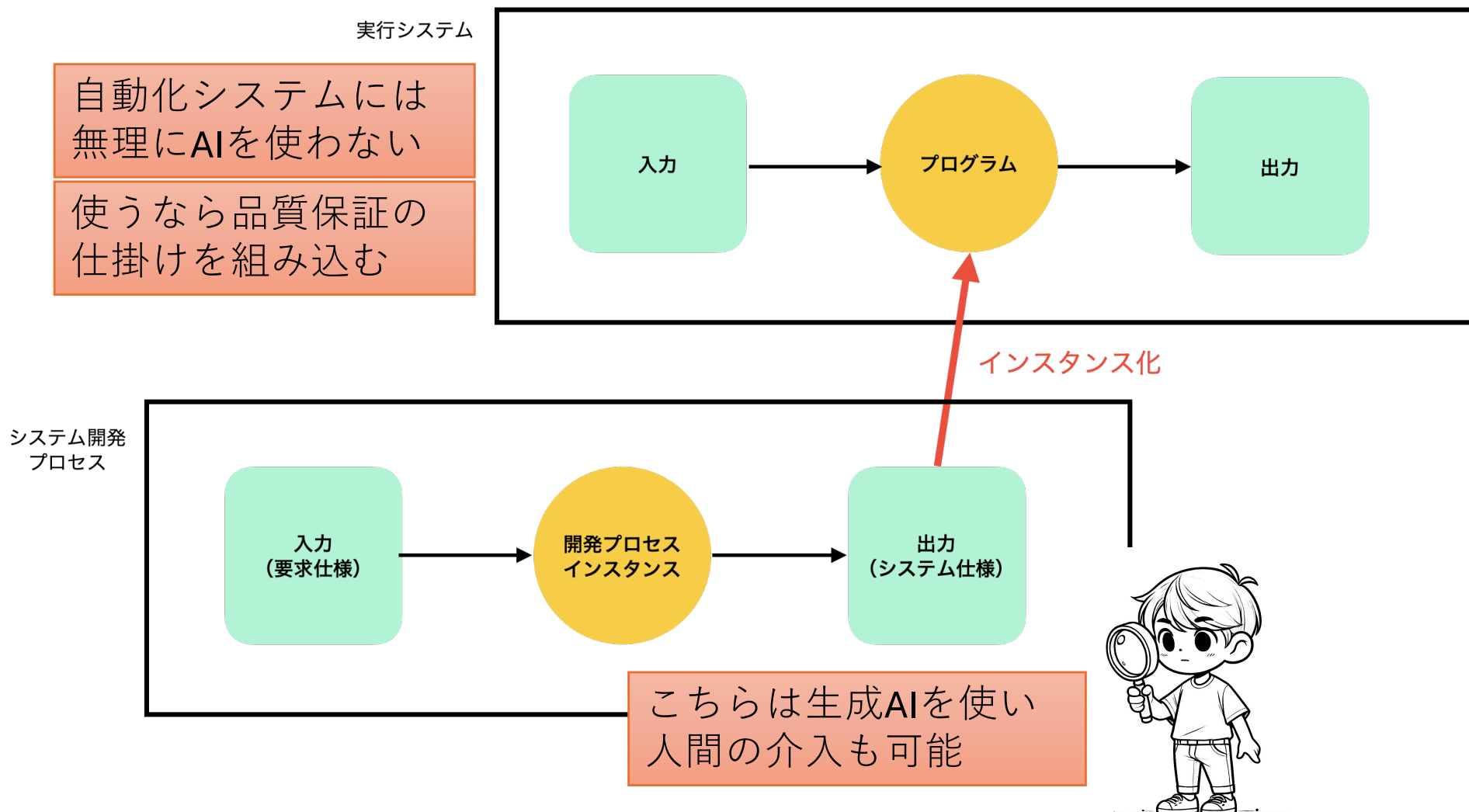
制約条件：もし品質が重要だとしたら

人間の介入が必要



高度な自動化システムには（まだ）不向き

# 以下は最終システムではなく、開発プロセスに生成AIを使うという話



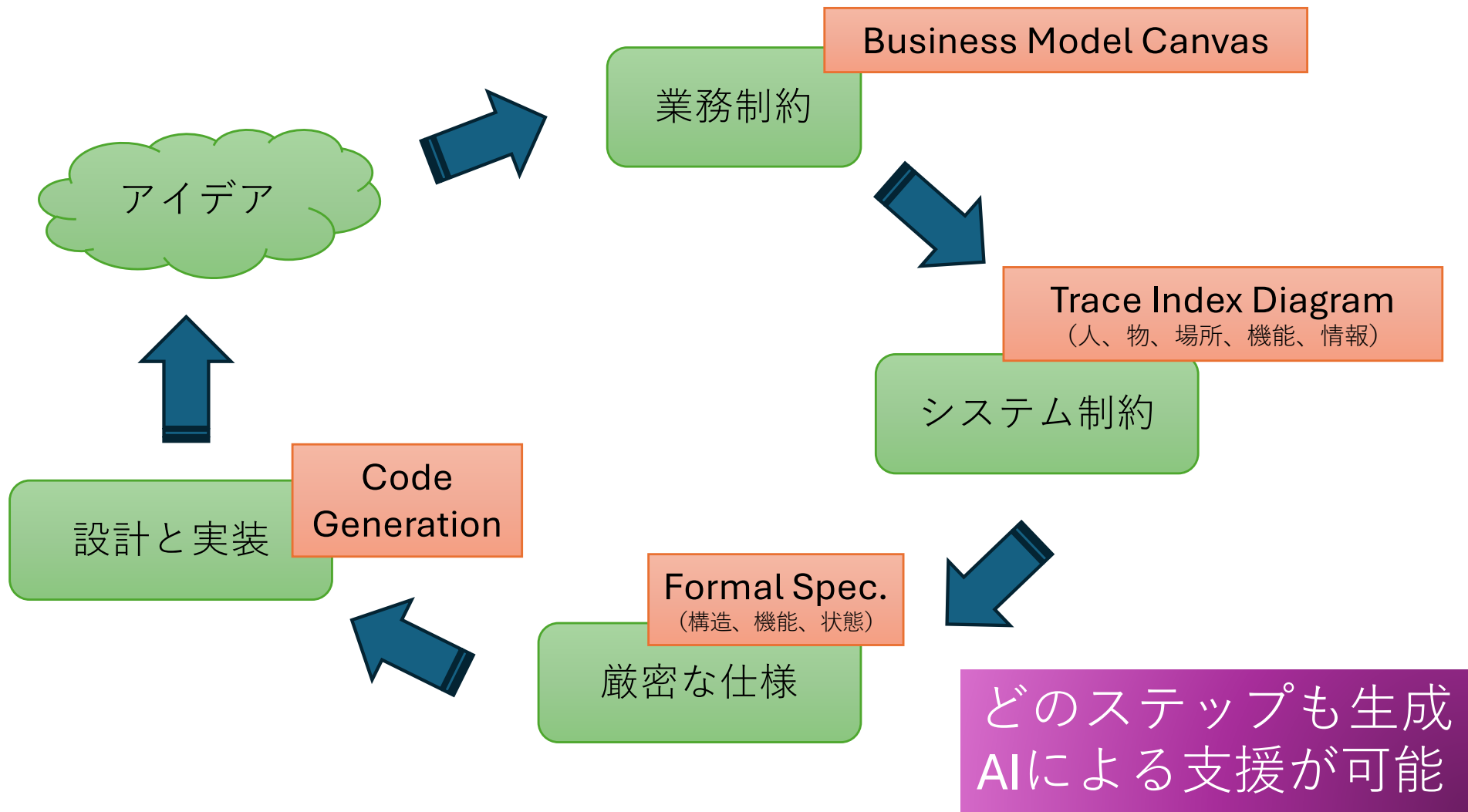
# ソフトウェア開発の ライフサイクル

上流工程への生成AIの応用

# ご注意

- この講演では、一種の開発方法論のようなものを例題として紹介します
- しかし本講演の**目的**はその使っている開発方法論そのものの紹介ではありません
- 目的は「自分で定義した開発方法論」を、どのように生成AIに手助けして貰うかです

# 想定するソフトウェア開発のライフサイクル

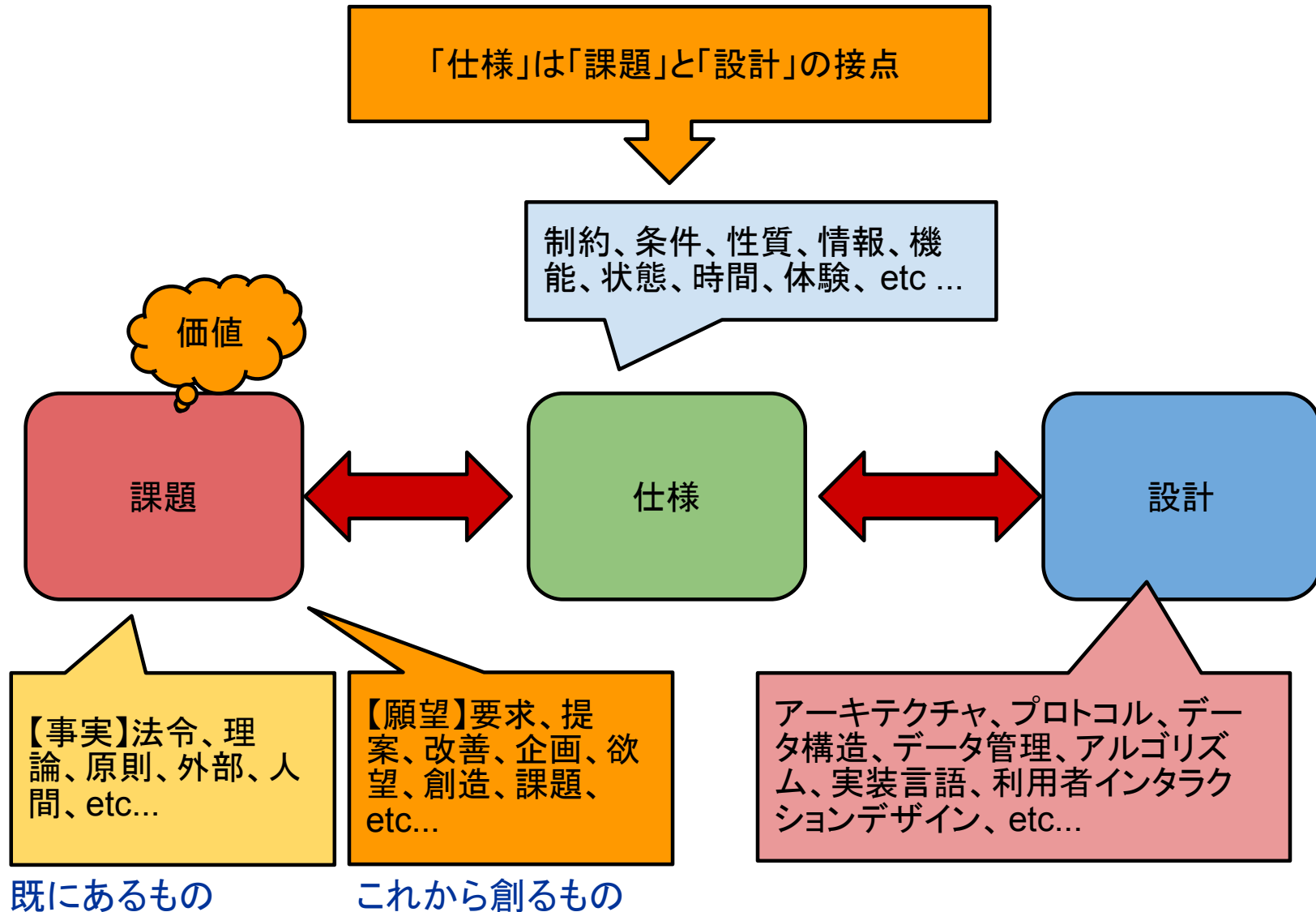




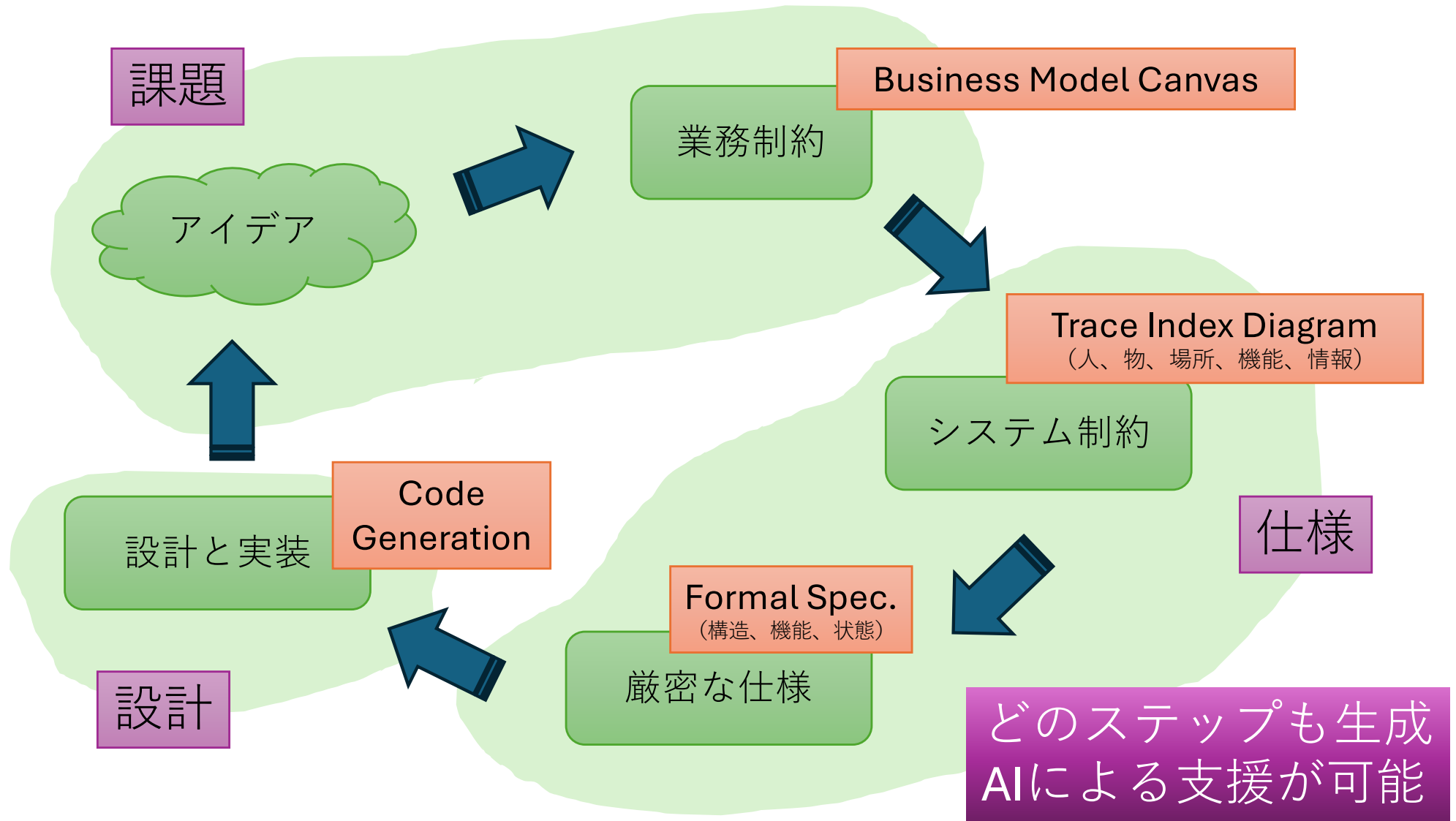
# システム開発の3つの視点

- 課題
  - **利用者の視点**
    - 現実世界にどのような問題を抱えており、そのうちどの部分をシステム化して解決したいと思っているのか
  - 入力: 問題領域知識(事実) + 業務要求(願望)
- 仕様
  - **利用者と開発者の視点**
    - 上で挙げられた課題の解決を、どのようなシステムで支援するのか
  - 入力: 課題 + システム要求
- 設計
  - **開発者の視点**
    - 要求されるシステムの仕様がどのように設計すべきか
  - 入力: 仕様 + 最新構築技術

# 課題・仕様・設計の要素



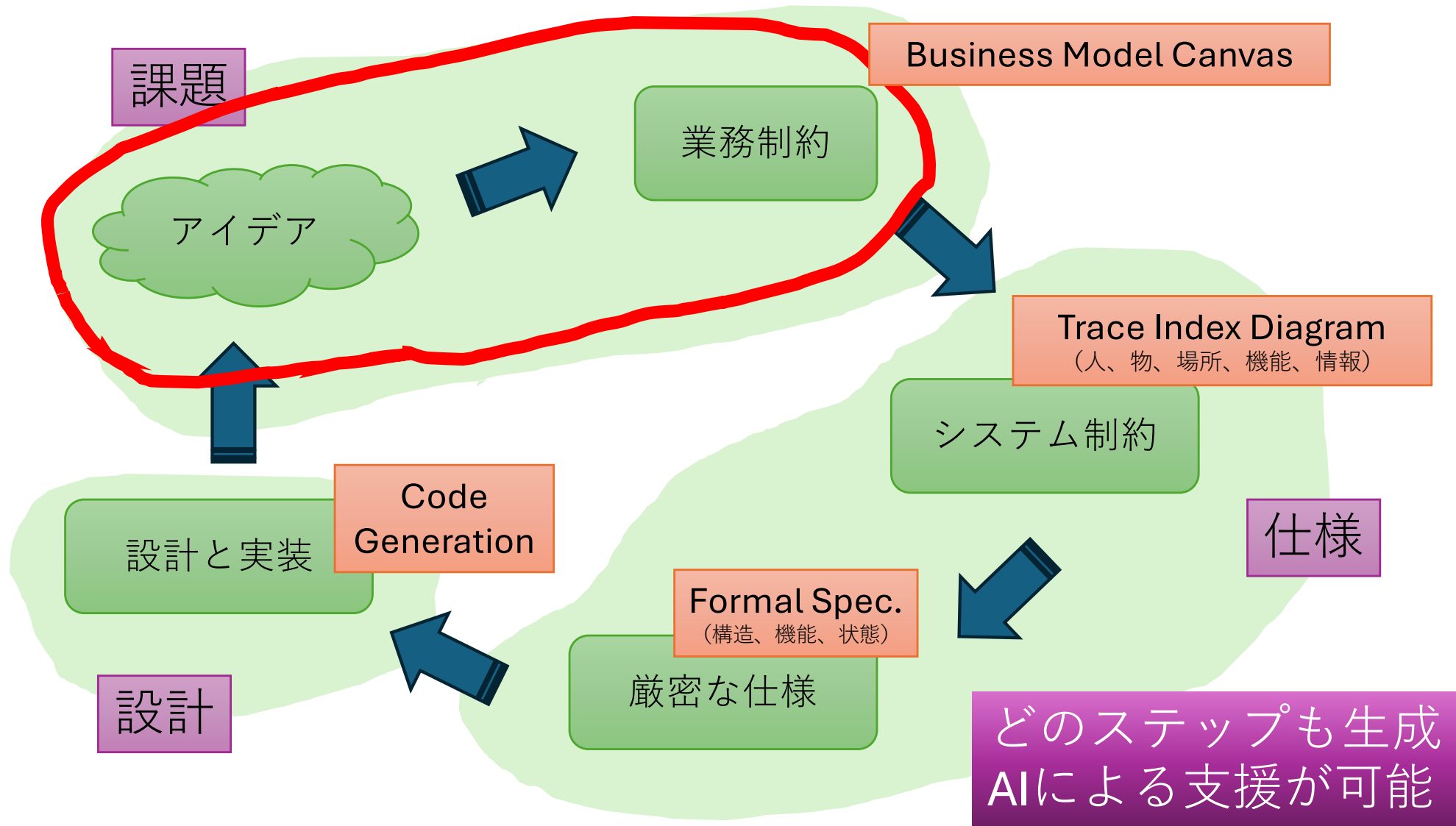
# ソフトウェア開発のライフサイクル



# アイデアから **BMC** へ

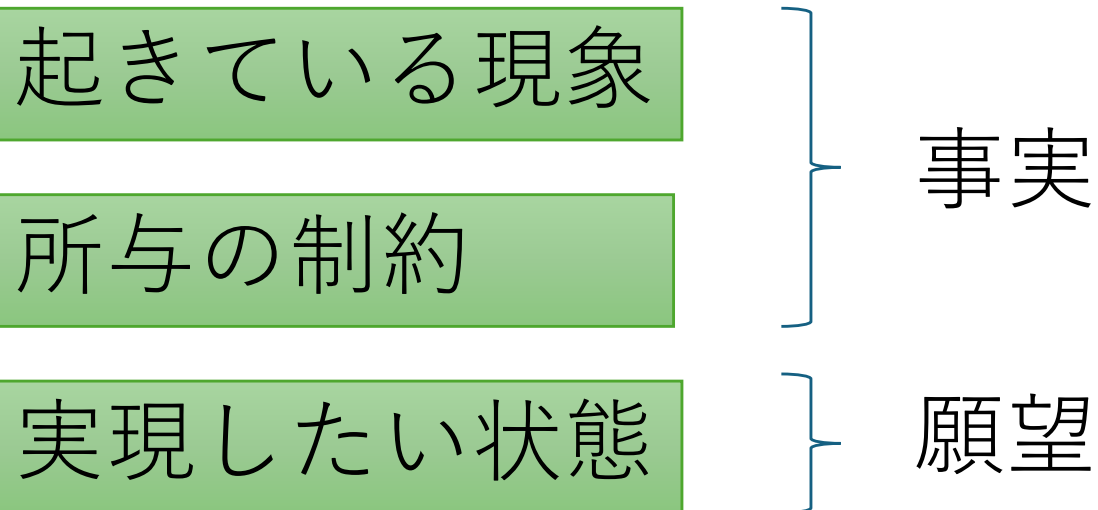
Ideas → Business Model Canvas

# ソフトウェア開発のライフサイクル



# 課題：アイデア・問題の発見

以下のような様々な視点を提出し、整理していく



最終的に実現したい**価値**は何か？

# ビジネスモデルキャンバス 1/2

- Customer Segments:
  - どの顧客や市場に価値を提供するか
- Value Propositions:
  - 顧客にどのような価値を提供するか
- Channels:
  - 価値提案を顧客にどのように届けるか
- Customer Relationships:
  - 顧客とどのような関係を築くか

# ビジネスモデルキャンバス 2/2

- **Revenue Streams:**
  - どのように収益を得るか
- **Key Resources:**
  - 価値提案を提供するために必要な主要な資源は何か
- **Key Activities:**
  - 価値提案を提供するために必要な主要な活動は何か
- **Key Partners:**
  - 価値提案を提供するために必要なパートナーシップは何か
- **Cost Structure:**
  - ビジネスモデルを実現するための主要なコストは何か



# アイデアからBMCへ

- ビジネスモデルキャンバス（BMC）の作成には、特定のコンテキストや環境に関する情報（事実や法則）と、そのビジネスモデルを通じて実現したい目標やビジョン（状態）の両方が必要です。
  1. **事実や法則**: これはビジネスが存在する環境や市場の実情を指します。たとえば、顧客のニーズや行動、競合の動向、規制や技術の変動など、ビジネスが直面する具体的な状況や条件を示す情報です。
  2. **実現したいもの（状態）**: これはビジネスの目標やビジョンを指します。たとえば、特定の市場でのリーダーシップの確立、新しい市場の開拓、特定の顧客セグメントへの対応など、ビジネスが達成したい成果や状態を示す情報です。
- これらの入力を元に、BMCの各セクション埋めていくことができます

# 例：アイデアからBMCへ

## ChatGPTへのプロンプト

- **事実や法則**：現在小さな町の図書館を運営しています。図書の出借や返却の手続きは、全て人間の職員が手作業で行っています。同じ本を複数冊蔵書として持っていることがあります。図書を借りられるのは図書館に会員登録をした町の住民だけです（住民登録済）一度に借りられる本は3冊までです。
- **実現したい状態**：貸出と返却処理を計算機を使って支援したい。実際の貸出、返却に際しては利用者が窓口で書籍を持参し、その書籍の情報を窓口の係員が読み取って処理が行われる。
- この入力から新しい図書館の貸出、返却に関する BMC を提案して下さい。

# 生成された BMC の素案

ChatGPTからの出力

提案するビジネスモデルキャンバス (BMC) の各セクションを以下の通りに作成します：

## 1. 顧客セグメント (Customer Segments)

1. 町の住民 (図書館に会員登録済み)

## 2. 価値提案 (Value Propositions)

1. 計算機を用いて迅速な貸出と返却処理
2. 複数冊の同じ本の管理が容易
3. 利用者が窓口での手続きを簡単に行える

## 3. チャネル (Channels)

1. 図書館の窓口

## 4. 顧客関係 (Customer Relationships)

1. 個人対個人のサービス (窓口での直接のサポート)

## 5. 収益の流れ (Revenue Streams)

1. このケースでは明確な収益の流れは述べられていませんが、例えば、延滞料や特定のサービス料などを考慮することができます。

## 6. 主要活動 (Key Activities)

1. 貸出・返却処理の自動化
2. 係員への新システムの研修

## 7. 主要リソース (Key Resources)

1. 計算機や関連するソフトウェア
2. 係員

## 8. 主要パートナーシップ (Key Partnerships)

1. 計算機やソフトウェアの供給業者

## 9. コスト構造 (Cost Structure)

1. 計算機やソフトウェアの導入・維持費
2. 係員の研修コスト

# Memo : Advanced Data Analysis (a.k.a Code Interpreter) を利用して図式化

ChatGPTからの出力

## 図書館の貸出・返却に関するビジネスモデルキャンバス

顧客セグメント	価値提案	チャンネル
<ul style="list-style-type: none"><li>・町の住民（図書館に会員登録済み）</li></ul>	<ul style="list-style-type: none"><li>・計算機を用いた迅速な貸出・返却</li><li>・複数冊の同じ本の管理</li><li>・簡単な窓口手続き</li></ul>	<ul style="list-style-type: none"><li>・図書館の窓口</li></ul>
顧客関係	コスト構造	収益の流れ
<ul style="list-style-type: none"><li>・個人対個人のサービス</li></ul>	<ul style="list-style-type: none"><li>・計算機・ソフトウェアの導入・維持</li><li>・研修コスト</li></ul>	<ul style="list-style-type: none"><li>・（例）延滞料</li><li>・特定のサービス料</li></ul>
主要活動	主要リソース	主要パートナーシップ
<ul style="list-style-type: none"><li>・貸出・返却の自動化</li><li>・新システムの研修</li></ul>	<ul style="list-style-type: none"><li>・計算機・ソフトウェア</li><li>・係員</li></ul>	<ul style="list-style-type: none"><li>・計算機・ソフトウェアの供給業者</li></ul>

# アイデアから**BMC**へ もっと大掛かりな仕掛け

売上が落ちている地元の商店街のメンバーから依頼を受けて、商店街活性化の施策を考えています。そのときに目指すべきビジネスモデルを**BMC**（ビジネスモデルキャンバス）で作成しようと思っています。

商店街のメンバー（顧客）を巻き込んで**BMC**を作成するための**プランを作成してください**。プランには必要なリソース、大まかなタイムラインなども含めてください。



[ChatGPT出力](https://chatgpt.com/share/67af16fa-3fd8-800b-ad0a-d465a0c7929b)

<https://chatgpt.com/share/67af16fa-3fd8-800b-ad0a-d465a0c7929b>

# アイデアから**BMC**へ もっと大掛かりな仕掛け

以下のようなプランで「地元商店街の活性化」に向けての**BMC**（ビジネスモデル・キャンバス）を作成しようとしています。このプランを実行した際に出力される可能性のある**BMC**（ビジネスモデルキャンバス）の例を3つ示してください。

==== プラン =====

前のスライドにあったプラン

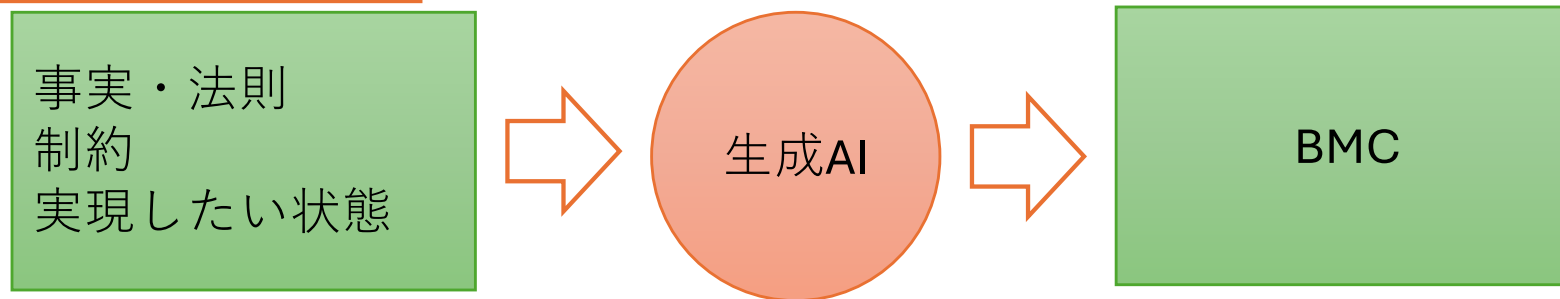


[ChatGPT出力](https://chatgpt.com/c/67af1375-1ce0-800b-9535-3f07fd028d25)

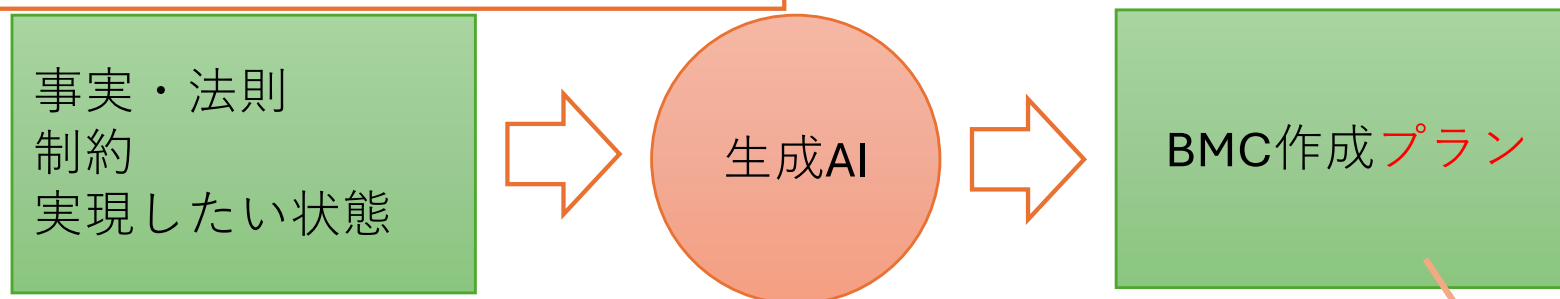
<https://chatgpt.com/c/67af1375-1ce0-800b-9535-3f07fd028d25>

# 2つの使い方例を示した

## 成果物を直接生成

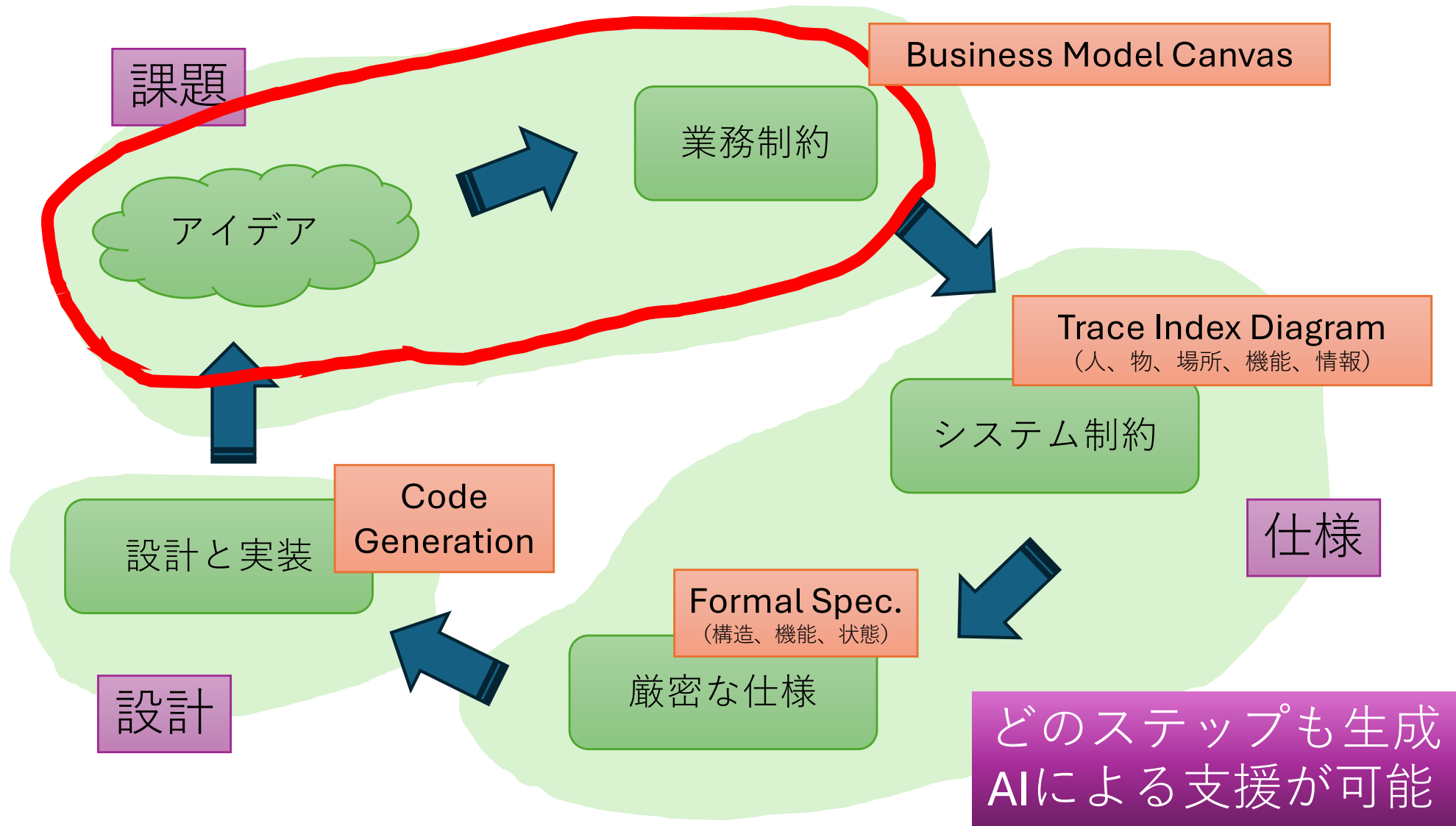


## 成果物を生成するプロセスを提案



このプランをもとに更に詳細なプランの作成や  
実行結果のまとめなどをAIの支援を受けながら  
行うことが可能

# ソフトウェア開発のライフサイクル

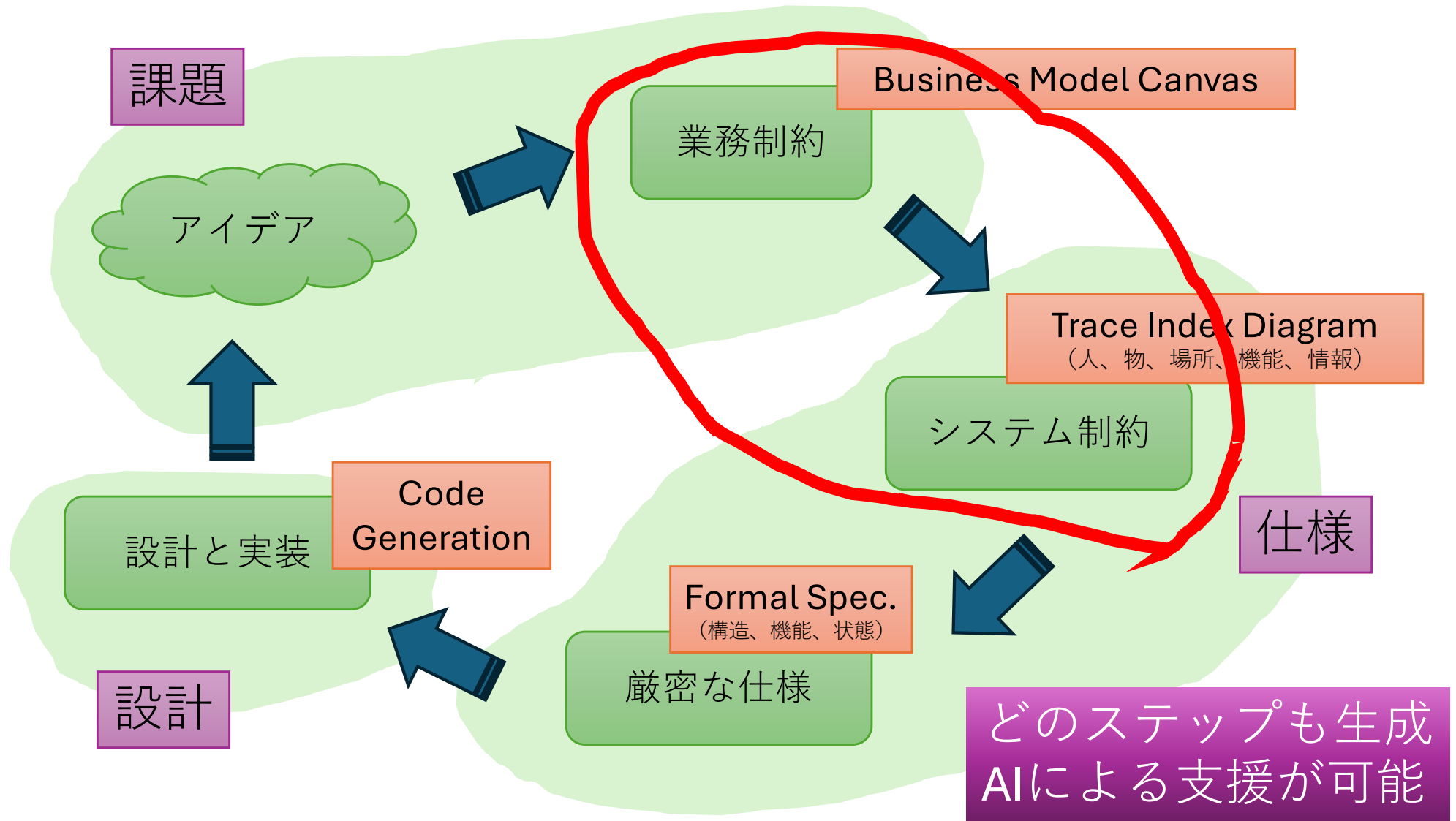




# BMC から TiD へ

Business Model Canvas → Trace index Diagram

# ソフトウェア開発のライフサイクル

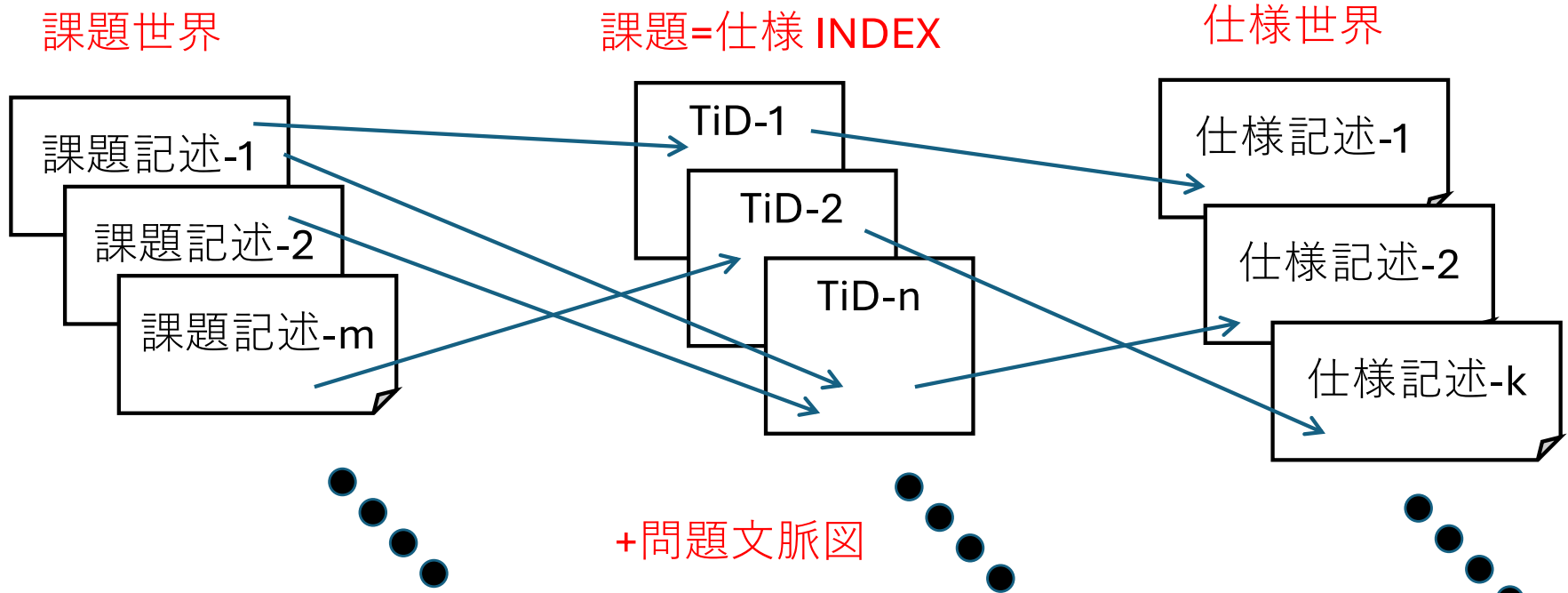


# TiD : 課題の世界から仕様の世界へ橋渡し

用語や条件の記述  
共通型、不変条件

業務シナリオと対応する仕様の記述  
イベント駆動型のシナリオ記述、仕様との対応関係

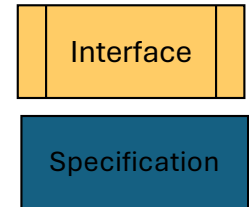
機能仕様記述  
情報型、関数、操作、状態



# TiD の主要構成要素

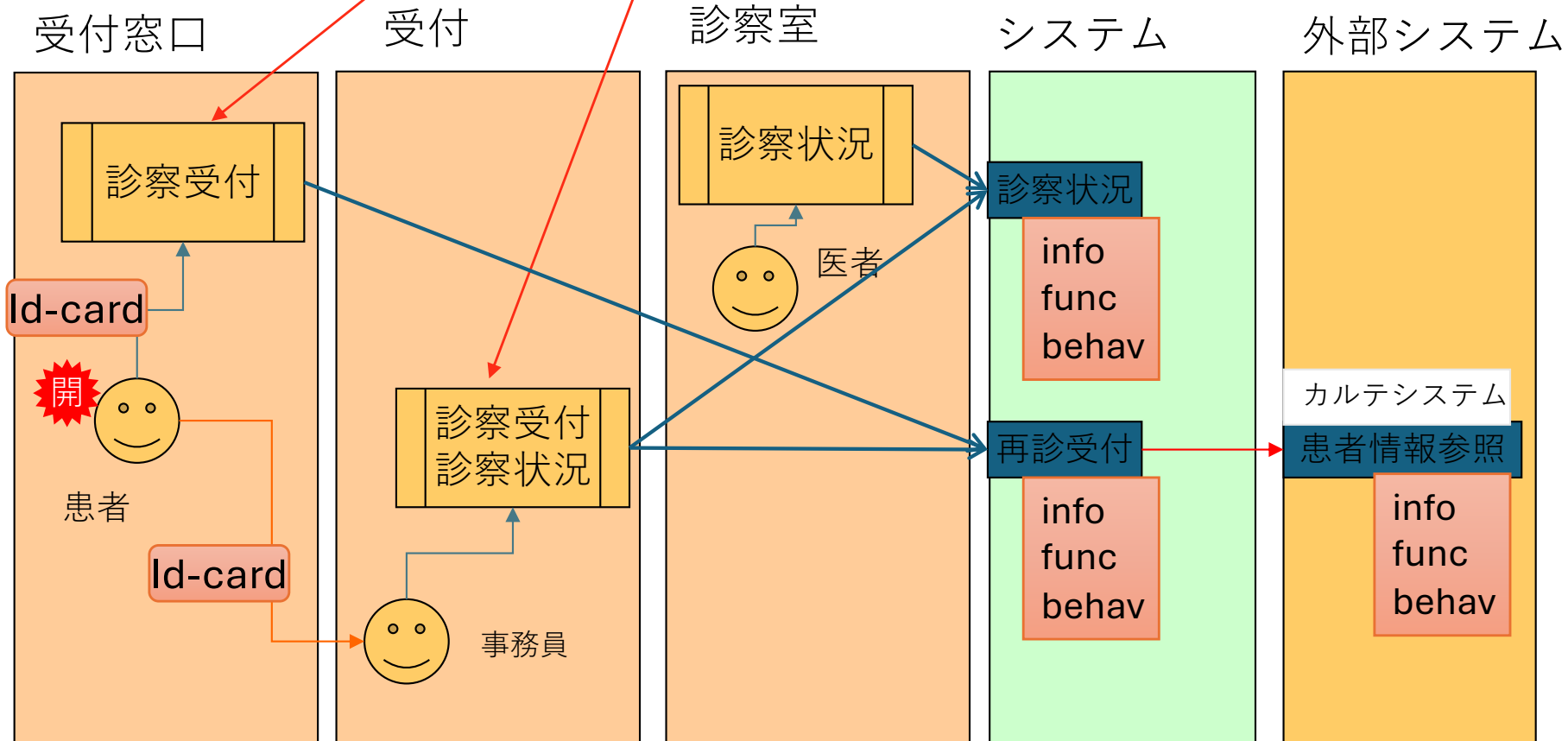
- **アクター**：サービスに関わる動作主体。人間や他のシステムなどで表現される。サービスの結果を受け取る役割もある。後述のようにアクターは特定のスペースの中に現れる。
- **インターフェイス**：アクターが使うサービスの界面。人間が使う画面や他システムが使うAPIなどがその例。後述のようにインターフェイスは特定のスペースの中に現れる。
- **スペース**：アクターとインターフェイスが置かれた場所。例えば「受付窓口」などがスペースの例。スペースではアクターとアクターが対話したり、アクターがインターフェイスを使ったりする。
- **ビジネスロジック**：インターフェイスや他のシステムが使う業務論理。システムに置かれている。ロジックを実現するために必要なデータモデルを操作する。データモデルは基本的にERモデルとして定義される。
- **データモデル**：ビジネスロジックを実現するために定義されるデータ構造。
- **システム**：インターフェイスや他のシステムが使うビジネスロジックが置かれた場所。ビジネスロジックが操作するデータモデルもシステムに置かれている。
- **外部システム**：システムに似ていて特定のビジネスロジックを提供するが、開発側からは基本的に変更することができない場所

# TiD 例(1): 同じ「受付」でも UseCase が異なると インターフェイスも異なる



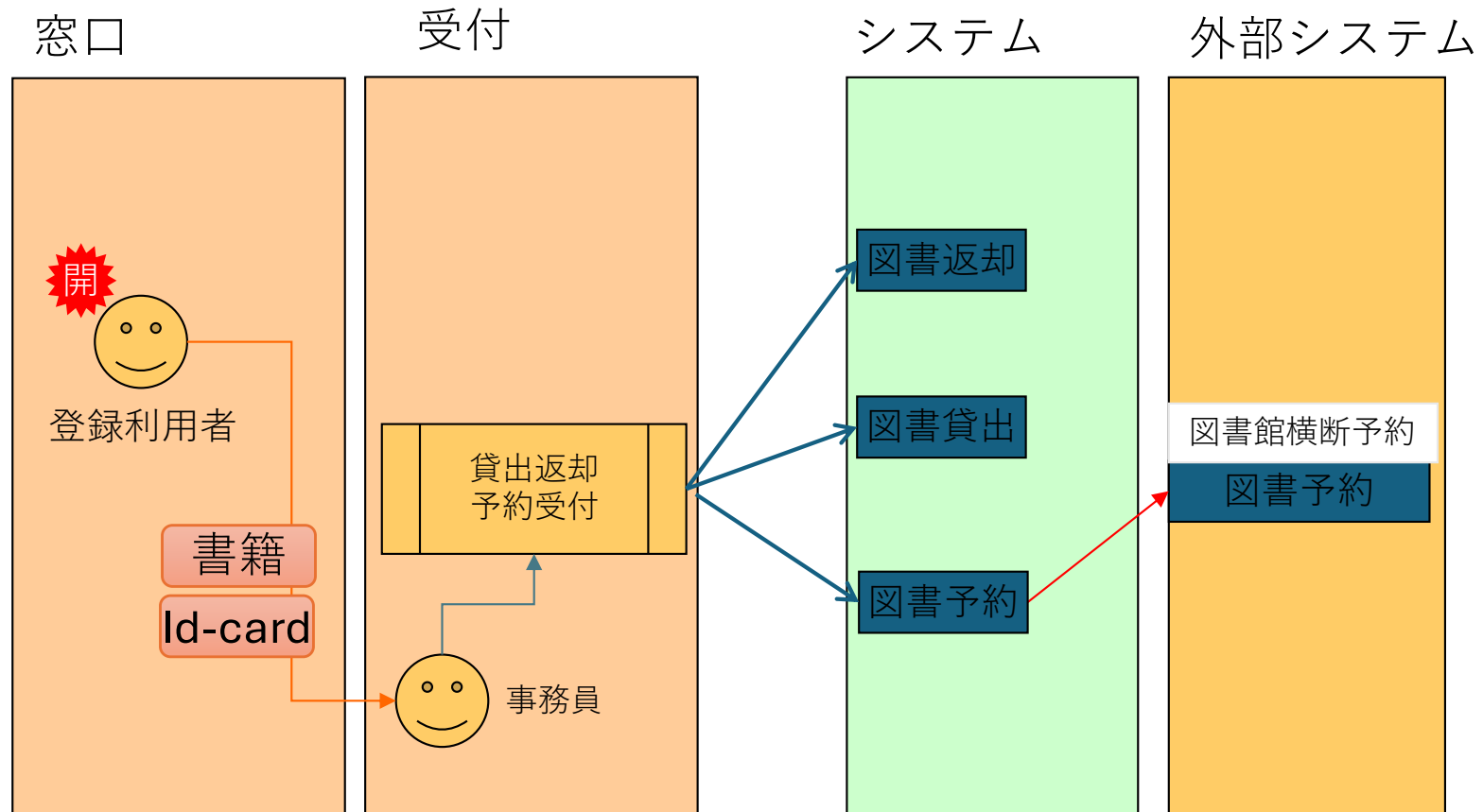
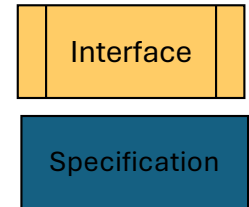
TiD  
「再診受付」

ICカード等による自動受付UseCase  
事務員による手動受付UseCase



# TiD 例(1)：図書館の予約、返却、予約

TiD  
「図書貸出・返却・予約」



# TiD の役割

- 課題世界が要求する一塊のシナリオを記述するためのダイアグラム（もしくはテキスト表現）である
- 一枚の TiD には**開始点**がある。開始点は特定のアクターの動作、もしくはその他の条件の成立（特定の日になるなど）によって決定される
- また TiD には**事後条件**が添えられている。開始点で始まった TiD が終了したときにはその事後条件が成立していることが求められる
- 複数のシナリオに対してそれぞれ TiD を記述し、最後に**全TiDを横断的に見る**ことで、システムを構成するアクター、インターフェイス、ビジネスロジック、データモデル、スペース、他システムの一覧を見ることができる

# BMC から TiD へ

## ChatGPTへのプロンプト

- それでは作成したビジネスモデルキャンバスからTiDに展開するためには、どのようなプロセスをとることができるでしょうか。ビジネスモデルキャンバスの内容が一通り埋められているとして、TiDに展開する際に、どのような点に着目しながらそれぞれのTiDを作成して行けば良いでしょうか。
- 出発点はビジネスモデルキャンバスの「主要活動」(Key Activities)とします。この際の基本的なガイドラインを示して下さい。



## 生成されたガイドライン 1/2

ビジネスモデルキャンバス（BMC）からTiDに展開する際の基本的なガイドラインを以下に示します：

### 1. 開始点の特定：

1. BMCの「主要活動」（Key Activities）を出発点とします。
2. これらの活動を具体的なシナリオやタスクに分解します。例えば、図書館のBMCでの「貸出・返却の自動化」は、具体的に「本の貸出」や「本の返却」といったシナリオに分解できます。

### 2. アクターの特定：

1. BMCの「顧客セグメント」や他のセクションで特定された利害関係者やシステムをアクターとして識別します。
2. それぞれのシナリオやタスクに関与する主要なアクターを明確にします。

### 3. インターフェイスの特定：

1. BMCの「チャンネル」や他のセクションから、アクターがサービスと対話するためのインターフェイスを特定します。

### 4. プレースの特定：

1. アクターとインターフェイスが相互作用する場所や環境を考えます。
2. シナリオやタスクの文脈に基づいて、具体的なプレースを識別します。

## 生成されたガイドライン 2/2

### 5. システムとビジネスロジックの特定：

1. 「主要リソース」や「主要パートナーシップ」から、シナリオやタスクをサポートするためのシステムやビジネスロジックを特定します。
2. これらのロジックは、具体的な操作や処理を示すものであり、データモデルとの相互作用を持つことがあります。

### 6. データモデルの特定：

1. ビジネスロジックが参照または操作するデータ構造やERモデルを明確にします。

### 7. TiDの連携：

1. 複数のTiDが存在する場合、それらがどのように連携するかを明確にします。例えば、あるTiDの終了が別のTiDの開始条件である場合などです。

### 8. 反復と評価：

1. 初めに作成したTiDを基に、システムやプロセスの全体像を確認します。
2. 各TiDを横断的に見て、矛盾や欠落している部分を特定し、必要に応じて修正や追加を行います。

BMCからTiDへの展開は、ビジネスの高レベルな概要からより具体的かつ実用的なシナリオへの移行をサポートします。このガイドラインは、その過程をスムーズに進めるための基本的なフレームワークを提供します。

# BMC to TiD

- この時点で、先のガイドラインをブラッシュアップして、顧客側と一緒に実際の **TiD** をワークショップ形式で作成しても良い
- もちろんこのときには **BMC** を参照し続ける
- 顧客とのワークショップを通して、**TiD** が記述されるだけでなく、**BMC** そのものにも色々とフィードバックがかかるはずである
- だがここでは、生成**AI**に対して最初のたたき台作成を依頼してみよう

# たたき台の作成

ChatGPTへのプロンプト

「本の貸出」「本の返却」に関する TiD をテキスト形式で生成して下さい。

# 生成された TiD (テキスト表現)

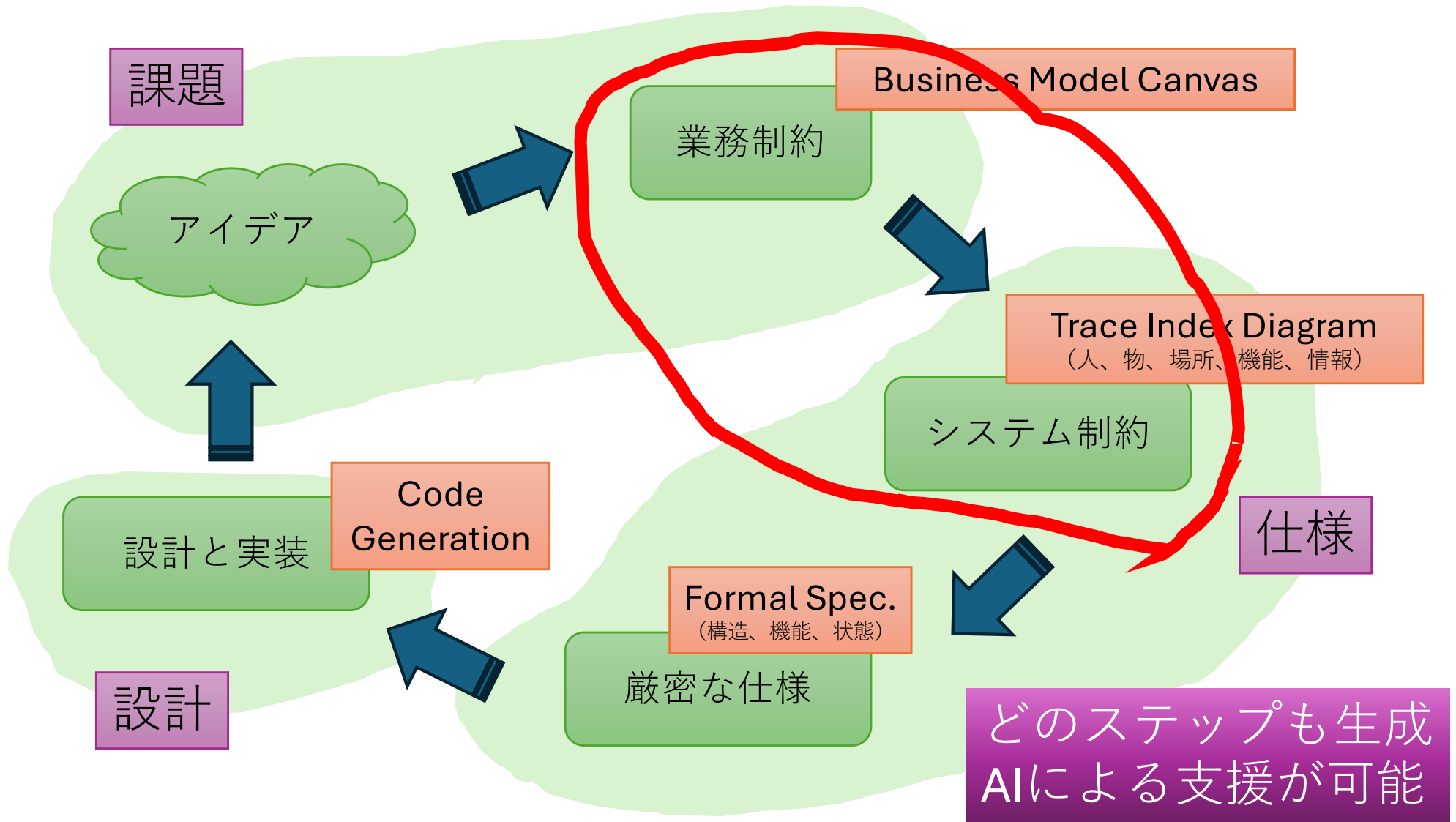
## TiD: 本の貸出

- **開始点:**
  - 会員登録済みの住民が図書館の窓口で本を借りる意向を示す。
- **アクター:**
  - 住民 (会員)
  - 窓口の係員
- **インターフェイス:**
  - 貸出端末
  - 本のバーコードリーダー
- **プレース:**
  - 図書館の貸出窓口
- **システム:**
  - 貸出管理システム
- **ビジネスロジック:**
  - 会員情報の確認
  - 貸出可能な本の確認
  - 貸出情報の登録
- **データモデル:**
  - 会員データ
  - 蔵書データ
  - 貸出履歴データ
- **事後条件:**
  - 貸出履歴データに新しいエントリーが追加される。
  - 蔵書データの該当する本のステータスが「貸出中」になる。
  - 住民 (会員) は借りた本を持って図書館を出る。

## TiD: 本の返却

- **開始点:**
  - 会員登録済みの住民が図書館の窓口で貸出中の本を返却する。
- **アクター:**
  - 住民 (会員)
  - 窓口の係員
- **インターフェイス:**
  - 返却端末
  - 本のバーコードリーダー
- **プレース:**
  - 図書館の返却窓口
- **システム:**
  - 貸出管理システム
- **ビジネスロジック:**
  - 会員情報の確認
  - 返却本の確認
  - 返却情報の更新
- **データモデル:**
  - 会員データ
  - 蔵書データ
  - 貸出履歴データ
- **事後条件:**
  - 貸出履歴データの該当エントリーが「返却済み」に更新される。
  - 蔵書データの該当する本のステータスが「利用可能」になる。
  - 住民 (会員) は返却した本を窓口に残して図書館を出る。

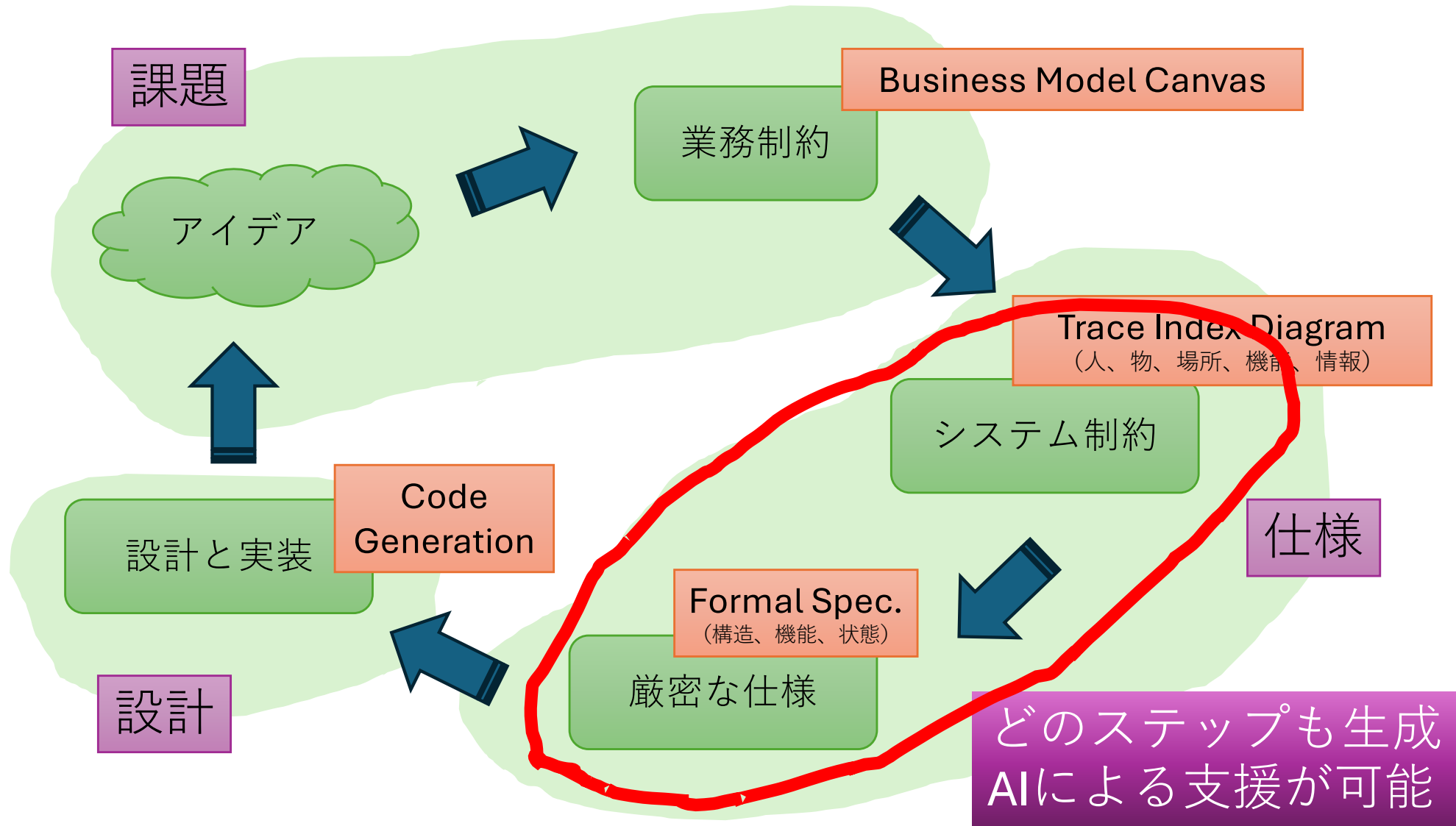
# ソフトウェア開発のライフサイクル



# 厳密な仕様の策定

形式仕様記述言語の利用

# ソフトウェア開発のライフサイクル





# TiD からデータモデルを導出

ChatGPTへのプロンプト

ではここで、上の2つのTiDに必要なデータモデルを定義して下さい。

仕様記述方法は VDM-SL に準拠するものとします。

# TiD からデータモデルを導出

```
types

Member ::
 memberId : nat
 name : seq of char
 borrowedBooks : set of Book;

Book ::
 bookId : nat
 title : seq of char
 status : BookStatus;

BookStatus = <AVAILABLE> | <BORROWED>;

BorrowRecord ::
 memberId : nat
 bookId : nat
 borrowDate : Date
 returnDate : [Date]; -- Optional date, as it might be null if not returned yet.

Date ::
 day : nat1
 month : nat1
 year : nat;

values

members : map nat to Member = {}->;
books : map nat to Book = {}->;
borrowRecords : set of BorrowRecord = {};
```

## ChatGPTからの出力

- ここで導出されたデータモデルは、途中あまり希望や制約を指定してこなかったため、生成AI側がある程度「想像」を行って生成したものになっている。
- 途中の対話に適宜介入することで、もっと細かく最終的な提案モデルをコントロールすることができる

# TiD からデータモデルを導出 (データ作成を依頼)

## ChatGPTへのプロンプト

以下のタイトルの書籍を初期値として持っているようなデータ定義を作成して下さい。

---

3月のライオン  
風と共に去りぬ  
山月記  
明日の記憶  
チーム・バチスタの栄光  
オブジェクト指向入門

## ChatGPTからの出力

types

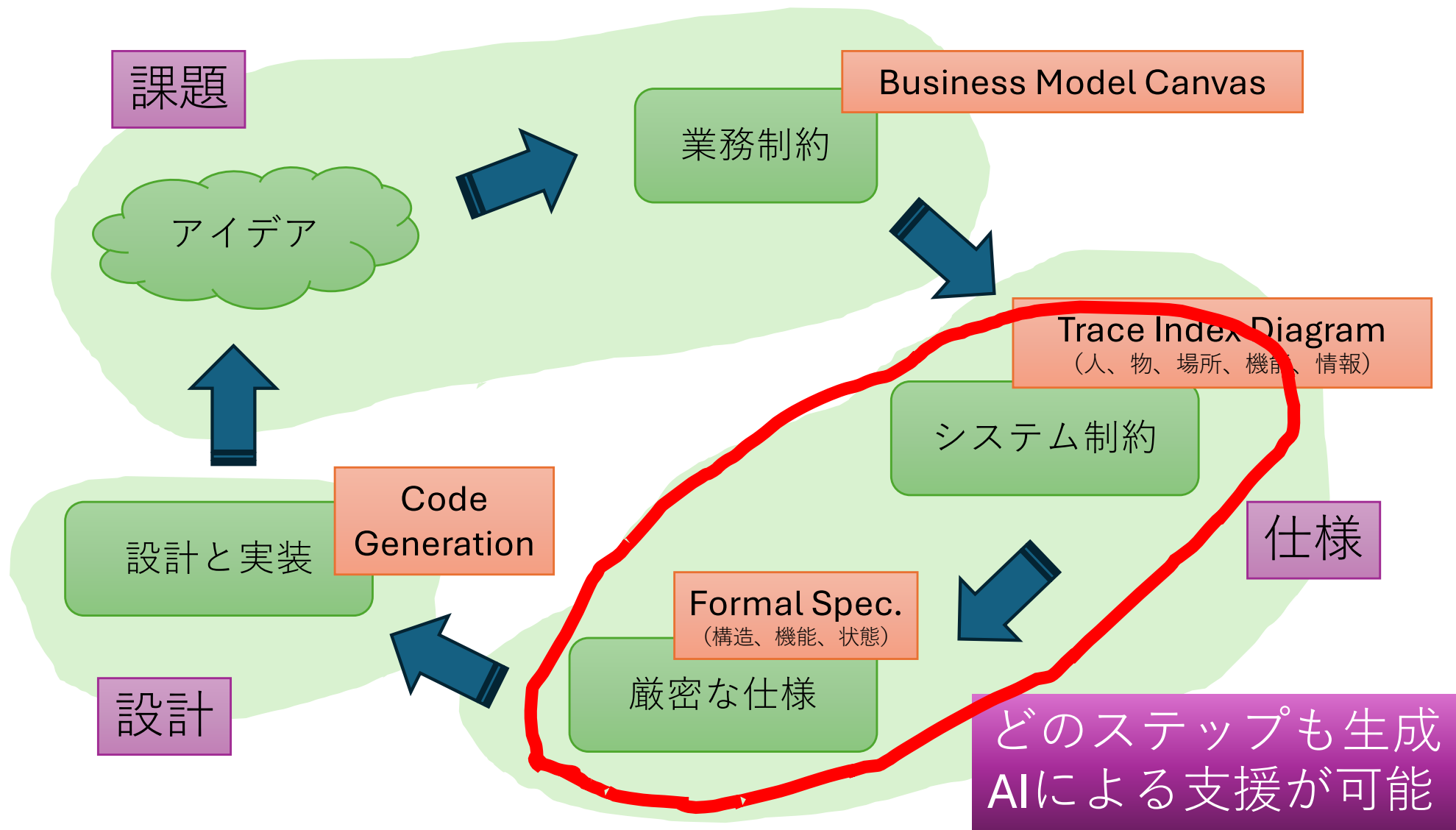
```
Book ::
 bookId : nat
 title : seq of char
 status : BookStatus;
```

```
BookStatus = <AVAILABLE> | <BORROWED>;
```

values

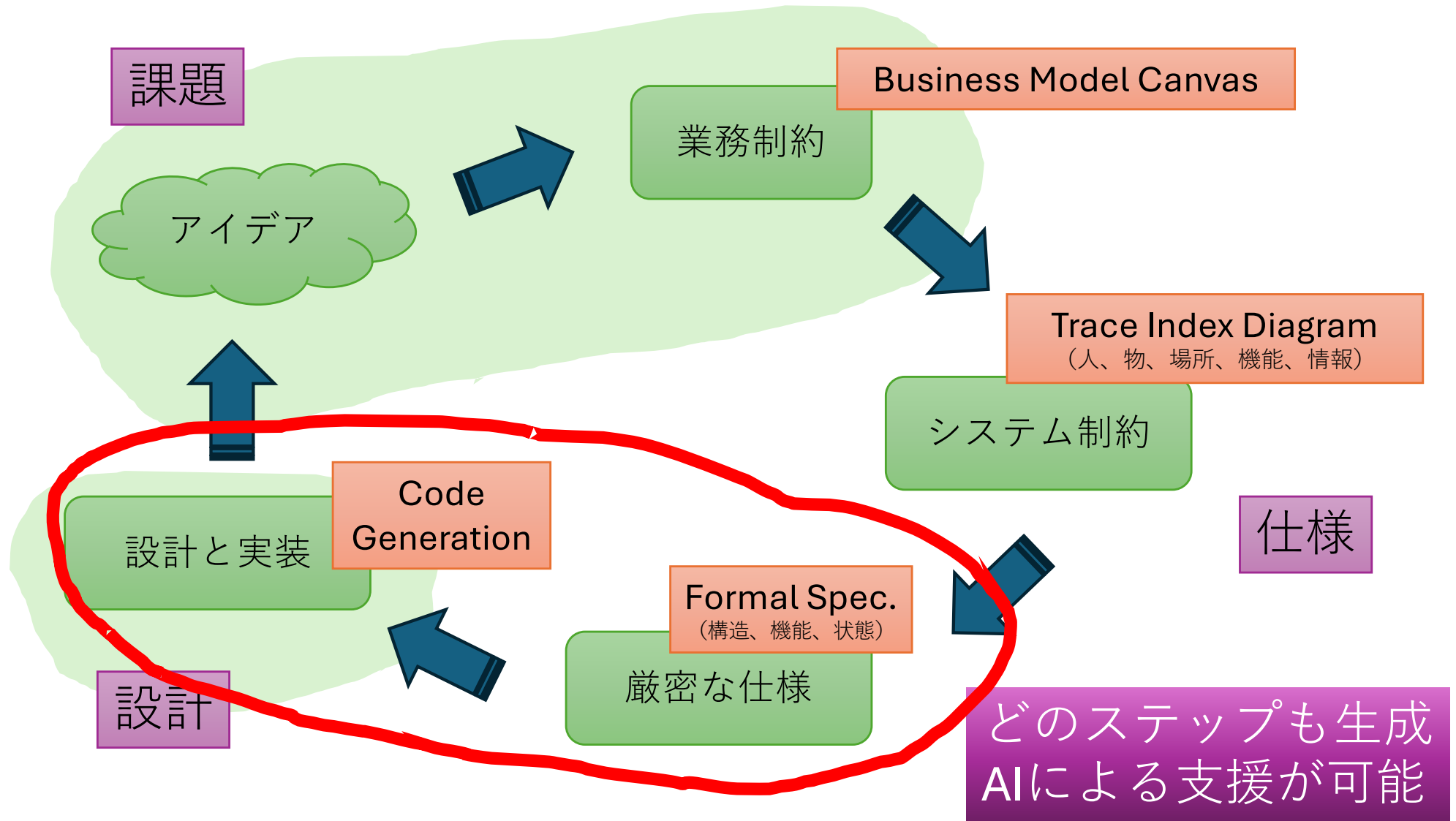
```
books : map nat to Book = {
 1 |-> mk_Book(1, "3月のライオン", <AVAILABLE>),
 2 |-> mk_Book(2, "風と共に去りぬ", <AVAILABLE>),
 3 |-> mk_Book(3, "山月記", <AVAILABLE>),
 4 |-> mk_Book(4, "明日の記憶", <AVAILABLE>),
 5 |-> mk_Book(5, "チーム・バチスタの栄光", <AVAILABLE>),
 6 |-> mk_Book(6, "オブジェクト指向入門", <AVAILABLE>)
};
```

# ソフトウェア開発のライフサイクル



# 仕様から実装へ

# ソフトウェア開発のライフサイクル



以下のように VDM-SL で書かれた putinbox の仕様を、仕様をみて設計を行う技術者向けに説明して下さい。

---

types

```
message = seq of char;
attachment = seq of char;
mail = message * seq of attachment;
inbox = seq of mail;
```

functions

```
putinbox : inbox * mail -> inbox
putinbox(ib, m) == is not yet specified
pre true
post
 if (forall a in set elems m .# 2 & isSafe(a)) then
 RESULT = ib ^ [m]
 else
 RESULT = ib;
```

```
isSafe : attachment -> bool
```

```
isSafe(a) == is not yet specified;
```

# ChatGPTによる解説

このVDM-SL仕様には、電子メールの基本的なモデリングが含まれています。以下に、この仕様の各部分を詳しく説明します。

## 型

1. **message**: 文字のシーケンスとしてのメッセージを表す型です。
2. **attachment**: 文字のシーケンスとしての添付ファイルを表す型です。
3. **mail**: メッセージと添付ファイルのシーケンスのペアを表す型です。この型は1つのメッセージと、それに関連する0またはそれ以上の添付ファイルを含むメールをモデル化します。
4. **inbox**: メールのシーケンスとしての受信箱を表す型です。

## 関数

1. **putinbox**: この関数は、受信箱 (**inbox**) とメール (**mail**) を引数として受け取り、メールが安全な場合にそのメールを受信箱に追加するものです。
  1. 前提条件 (**pre**): この関数は特に前提条件がありません (常にtrue)。
  2. 事後条件 (**post**): メールすべての添付ファイルが安全である場合 (**isSafe(a)**がtrueである場合)、メールは受信箱の末尾に追加されます。そうでない場合、受信箱は変更されません。
2. **isSafe**: この関数は、添付ファイル (**attachment**) を引数として受け取り、その添付ファイルが安全かどうかを判定します。具体的な判定基準や方法はこの仕様からはわかりません。プログラマーは、添付ファイルが安全かどうかを確認するための具体的な実装を提供する必要があります。

## 実装のための説明:

- この仕様に基づいてプログラムを実装する場合、**message** や **attachment** は、文字列や文字のリストとして実装できます。
- **mail** は、メッセージと添付ファイルのリストのペア (タプルや構造体) として実装できます。
- **inbox** はメールのリストまたは配列として実装します。
- **putinbox** 関数は、新しいメールを受信箱に追加するかどうかを判断するためのロジックを実装する必要があります。この判断は、添付ファイルの安全性に基づいています。
- **isSafe** 関数は、添付ファイルが安全かどうかを判定するための具体的なロジックや方法を実装する必要があります。この部分は、ウイルススキャンや他のセキュリティ検査を考慮するかもしれません。
- 全体として、この仕様は受信箱の基本的なモデリングを提供しますが、メールが受信箱に追加されるかどうかを判断するための詳細なロジックや、添付ファイルが安全かどうかを判断するロジックについては、プログラマーが具体的に実装する必要があります。



# 実装コードの生成

```
import java.util.ArrayList;
import java.util.List;
public class EmailSystem {

 public static void main(String[] args) {
 // Sample usage
 Inbox inbox = new Inbox();
 Mail mailWithSafeAttachment = new Mail("Hello!",
List.of("safe_attachment.txt"));
 Mail mailWithUnsafeAttachment = new Mail("Alert!", List.of("virus.exe"));

 inbox.putInbox(mailWithSafeAttachment);
 System.out.println(inbox); // Expected: [Mail[message=Hello!,
 attachments=[safe_attachment.txt]]

 inbox.putInbox(mailWithUnsafeAttachment);
 System.out.println(inbox); // Expected: [Mail[message=Hello!,
 attachments=[safe_attachment.txt]]
 }

 static class Inbox {
 private List<Mail> mails;

 public Inbox() {
 this.mails = new ArrayList<>();
 }

 public void putInbox(Mail mail) {
 boolean allAttachmentsSafe =
 mail.attachments.stream().allMatch(Attachment::isSafe);
 if (allAttachmentsSafe) {
 mails.add(mail);
 }
 }

 @Override
 public String toString() {
 return mails.toString();
 }
 }
}
```

```
static class Mail {
 private String message;
 private List<String> attachments;

 public Mail(String message, List<String> attachments) {
 this.message = message;
 this.attachments = attachments;
 }

 @Override
 public String toString() {
 return "Mail[message=" + message + ", attachments=" + attachments + "];"
 }
}

static class Attachment {
 public static boolean isSafe(String attachment) {
 // Sample logic: any attachment named "virus.exe" is unsafe.
 return !attachment.equals("virus.exe");
 }
}
}
```

# 仕様に関する質問（開発者向け）

この仕様の下で、添付ファイルのないメールを受信箱に追加しようとするとうようになりますか？

この仕様において、添付ファイルのないメール（すなわち、添付ファイルのシーケンスが空であるメール）を受信箱に追加しようとした場合の動作は、`putinbox`関数の事後条件（`post`）に基づいて判断されます。

事後条件を確認すると、次のように記述されています：

```
sql Copy code

if (forall a in set elems m. # 2 & isSafe(a)) then
 RESULT = ib^ [m]
else
 RESULT = ib;
```

この条件に基づいて詳しく解釈します。

1. `forall a in set elems m. # 2 & isSafe(a)`
  - この部分は、メール`m`のすべての添付ファイル`a`に対して、`isSafe(a)`が`true`であることを確認します。
2. メールが添付ファイルを持たない場合（添付ファイルのシーケンスが空の場合）、上記の条件は真になります。なぜなら、真偽の判定を行う対象の添付ファイルが存在しないため、条件は真を返します。
3. したがって、添付ファイルのないメールは、事後条件の`RESULT = ib^ [m]`部分に一致します。これは、メール`m`を受信箱`ib`の末尾に追加することを意味します。

結論として、この仕様の下で、添付ファイルのないメールを受信箱に追加しようとする、そのメールは受信箱の末尾に正常に追加されます。

# 仕様に関する質問（利用者向け）

この仕様の下で、添付ファイルのないメールを受信箱に追加しようとするとうどうなりますか？

なお開発者ではなく利用者の視点で答えて下さい。

了解しました。利用者の視点からの答えを以下に示します。

この仕様の下で、添付ファイルのないメールを受信箱に追加しようとする場合、そのメールは問題なく受信箱に追加されます。添付ファイルが安全かどうかの確認は必要ありませんので、追加の手間や待機時間なく、そのメールは受信箱の一覧に表示されることになります。

# エージェント型開発環境の台頭

他にもたくさんありますが... 例として。ファイルの生成やデバッグもアシスト

- **Roo Code (Cline からの派生)**
  - Roo Codeは、Visual Studio Code (VSCode)で利用可能なAIによるコーディングアシスタントの拡張機
  - 以前はRoo-Clineという名称だったが、v3.2以降でRoo Codeに改称された
- **Cursor**
  - Cursorは、AIを活用した次世代のコードエディタで、開発効率を大幅に向上させる革新的なツール
  - VS Codeをベースに開発されており、AIとの自然な対話を通じてコーディングを支援する
- **Windsurf**
  - やはりVSCodeからのフォーク。非常に大規模なコードベースを理解してAI支援の編集を行うことができる

# エージェント型開発環境の特徴

- 主な特徴
  - 自然言語によるコード生成と編集機能
  - マルチモーダル対応（テキスト、画像、音声入力）
  - プロジェクト全体の文脈理解と自動コード最適化
  - クラウドとローカル環境のシームレスな統合
  - 複数のプログラミング言語に対応（Python、JavaScript、Goなど）
- 主要機能
  - コードのリファクタリングと自動ドキュメント生成
  - AIによるテストケースの自動作成と実行
  - リアルタイムコラボレーション機能
  - バージョン管理システムとの統合
  - カスタマイズ可能なコード生成テンプレート
- 活用メリット
  - 開発速度の大幅な向上と作業効率化
  - チーム間のコミュニケーション改善と知識共有
  - コードの品質と一貫性の向上
  - 初心者から上級者まで幅広いスキルレベルに対応
  - 既存のツールチェーンとのシームレスな連携
- 利用上の注意点
  - セキュリティとプライバシーに関する設定を確認する必要がある
  - 人間によるレビューが重要
  - 大規模プロジェクトでは処理速度に影響する可能性がある
  - 生成されたコードの理解と検証が不可欠

# Roo Code の例

プロンプト

画面から入力した整数 N がコラッツ予想に従うか否かを確認するウェブアプリケーションを作成してください。  
1000回以上計算を繰り返しても予想が正しいかどうかわかないときは「不明」と表示してください。HTML、CSS、JavaScript を用いて1ファイルで作成してください。



Webアプリの画面

## コラッツ予想チェッカー

整数を入力してください:

27 はコラッツ予想に従います。(111 回) 計算過程: 27 -> 82 -> 41 -> 124 -> 62 -> 31 -> 94 -> 47 -> 142 -> 71 -> 214 -> 107 -> 322 -> 161 -> 484 -> 242 -> 121 -> 364 -> 182 -> 91 -> 274 -> 137 -> 412 -> 206 -> 103 -> 310 -> 155 -> 466 -> 233 -> 700 -> 350 -> 175 -> 526 -> 263 -> 790 -> 395 -> 1186 -> 593 -> 1780 -> 890 -> 445 -> 1336 -> 668 -> 334 -> 167 -> 502 -> 251 -> 754 -> 377 -> 1132 -> 566 -> 283 -> 850 -> 425 -> 1276 -> 638 -> 319 -> 958 -> 479 -> 1438 -> 719 -> 2158 -> 1079 -> 3238 -> 1619 -> 4858 -> 2429 -> 7288 -> 3644 -> 1822 -> 911 -> 2734 -> 1367 -> 4102 -> 2051 -> 6154 -> 3077 -> 9232 -> 4616 -> 2308 -> 1154 -> 577 -> 1732 -> 866 -> 433 -> 1300 -> 650 -> 325 -> 976 -> 488 -> 244 -> 122 -> 61 -> 184 -> 92 -> 46 -> 23 -> 70 -> 35 -> 106 -> 53 -> 160 -> 80 -> 40 -> 20 -> 10 -> 5 -> 16 -> 8 -> 4 -> 2 -> 1

# Roo Code の例 (生成コード)

```
<!DOCTYPE html>
<html>
<head>
<title>コラッツ予想チェッカー</title>
<style>
body {
 font-family: sans-serif;
}
#result {
 margin-top: 20px;
 font-weight: bold;
}
</style>
</head>
<body>
<h1>コラッツ予想チェッカー</h1>
<label for="number">整数を入力してください:</label>
<input type="number" id="number" min="1">
<button onclick="checkCollatz()">チェック</button>
<p id="result"></p>

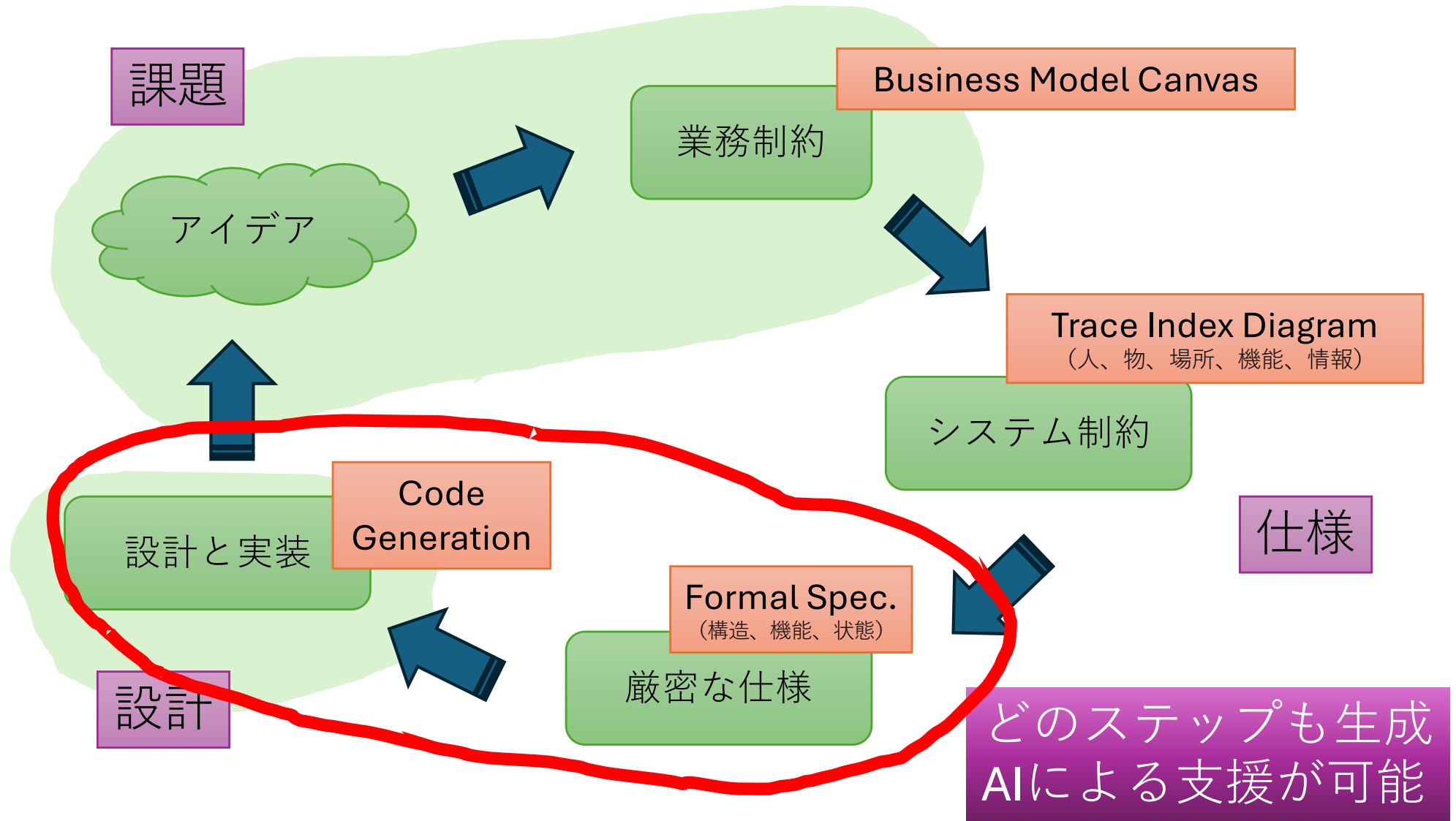
<script>
function checkCollatz() {
 let n = parseInt(document.getElementById("number").value);
 if (isNaN(n) || n < 1) {
 document.getElementById("result").textContent = "有効な整数を入力して
ください。";
 return;
 }

 let count = 0;
 while (n !== 1 && count < 1000) {
 if (n % 2 === 0) {
 n = n / 2;
 } else {
 n = 3 * n + 1;
 }
 count++;
 }
}
```

```
let steps = "";
let originalN = parseInt(document.getElementById("number").value);
let tempN = originalN;
while (tempN !== 1 && count < 1000) {
 steps += tempN + " -> ";
 if (tempN % 2 === 0) {
 tempN = tempN / 2;
 } else {
 tempN = 3 * tempN + 1;
 }
}
steps += tempN;

if (n === 1) {
 document.getElementById("result").textContent = originalN + " はコ
ラッツ予想に従います。 (" + count + " 回)¥n" + "計算過程: " + steps;
} else {
 document.getElementById("result").textContent = originalN + " は不明
(1000回以上計算しても1になりませんでした。)¥n" + "計算過程: " + steps;
}
}
</script>
</body>
</html>
```

# ソフトウェア開発のライフサイクル





# 検証

# 検証の二つの視点

- 正しいものを作っているか？ -- Validation
- 正しくものを作っているか？ -- Verification

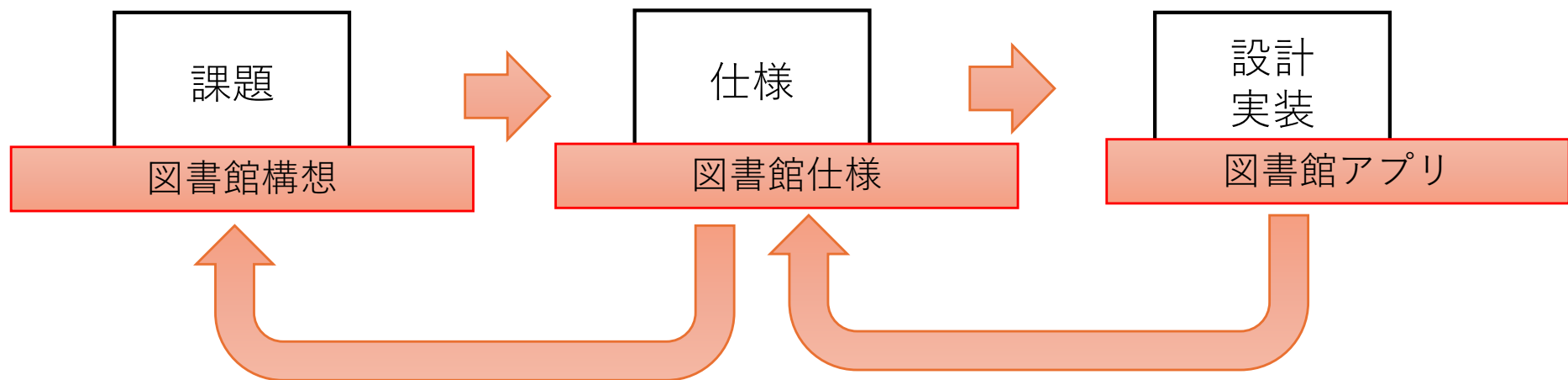
## 出典

B. W. Boehm. 1979. Guidelines for Verifying and Validating Software Requirements and Design Specifications. In Euro IFIP 79, P. A. Samet (Ed.). North Holland, 711–719.)

- Validation : Are we building the right product? (正しいものを作っているか?)
- Verification : Are we building the product right? (正しくものを作っているか?)

# 正しくものを作っているか？

## Verification



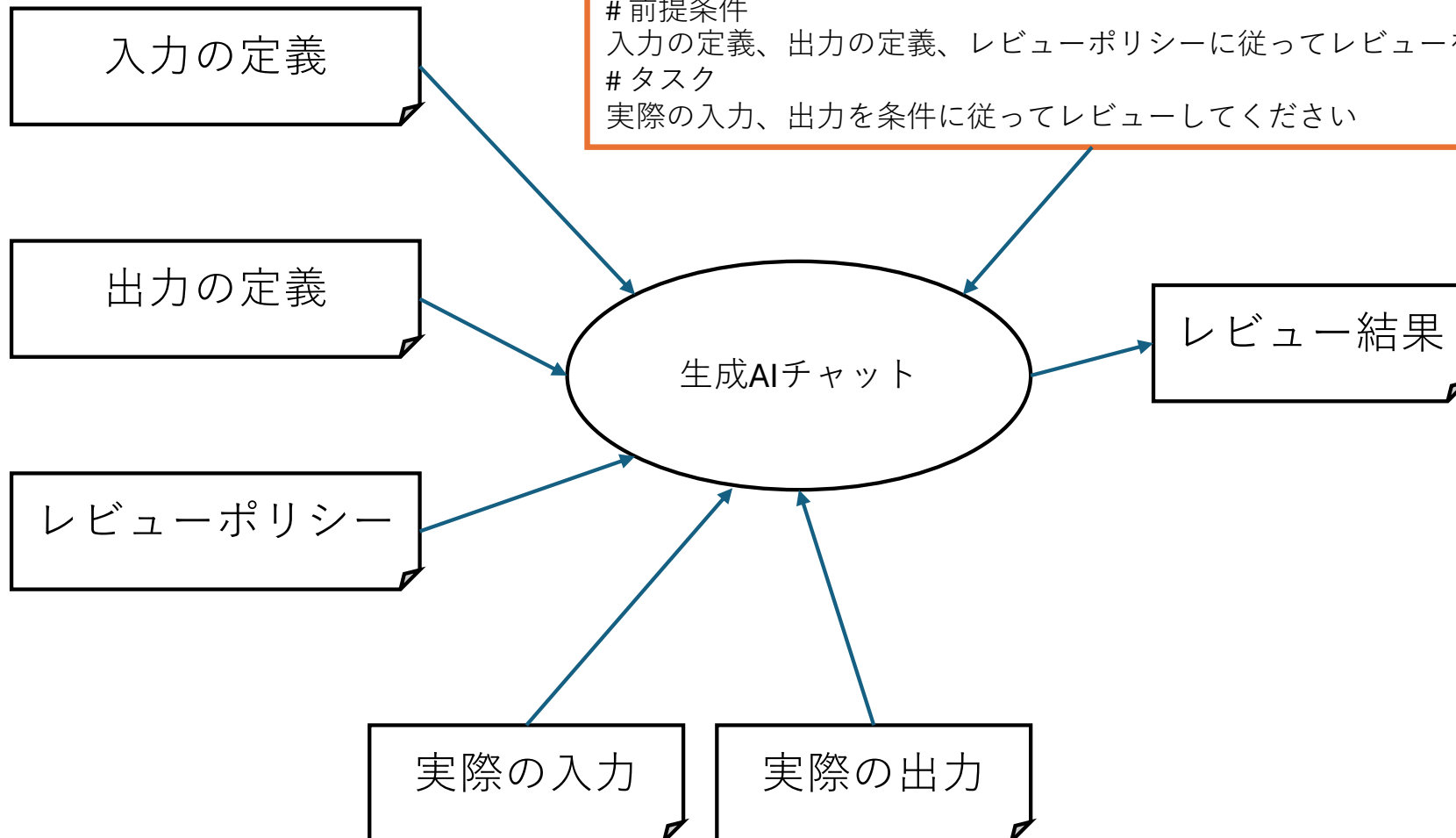
正しくものを作っているか？ 正しくものを作っているか？

テストとレビューで確認することが多い  
レビューもテストケース生成も生成AIを活用することができる  
その際に仕様が明確であれば効果が高い

# レビューと生成AI

## プロンプト

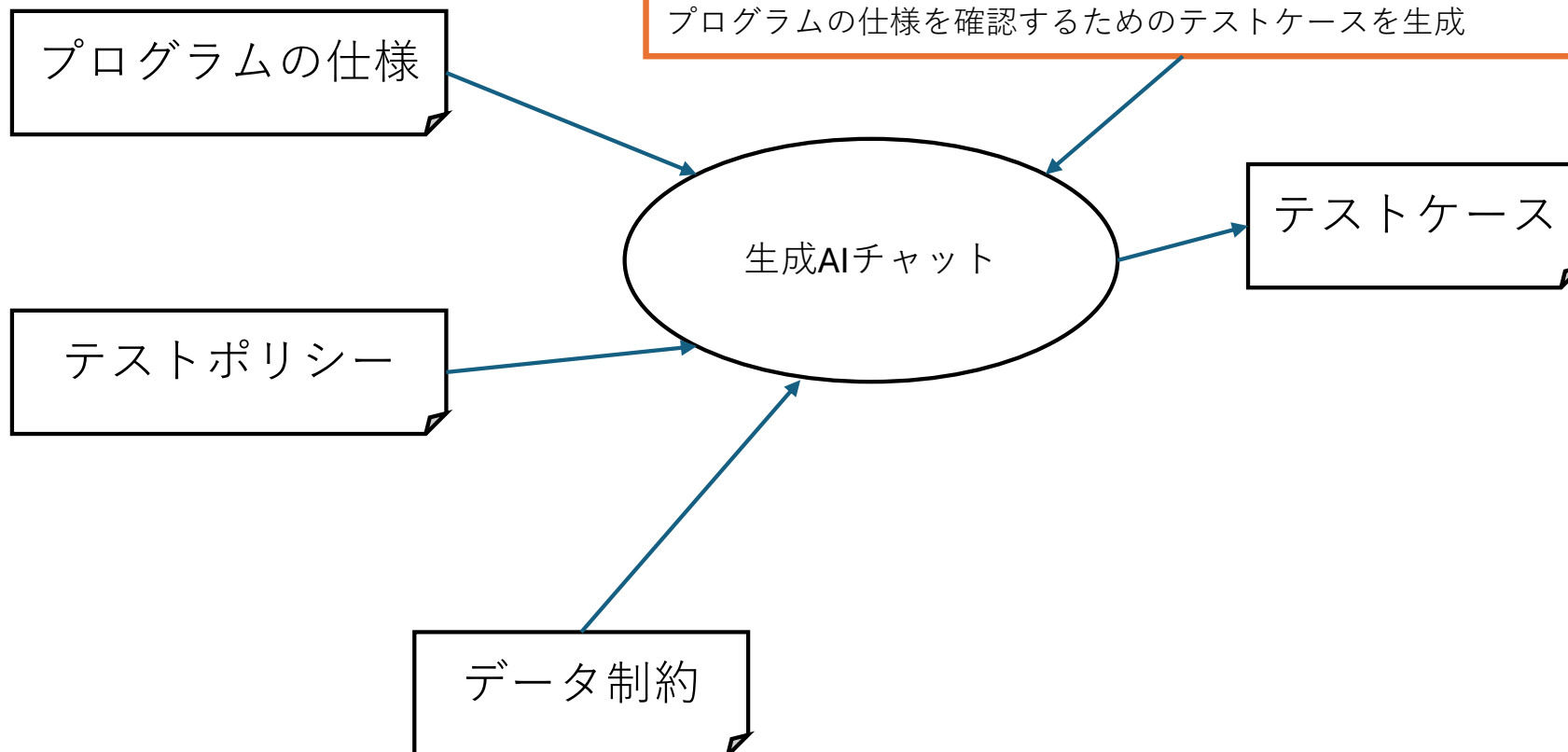
# 役割  
あなたは優秀なレビューラーです  
# 前提条件  
入力の定義、出力の定義、レビューポリシーに従ってレビューを行う  
# タスク  
実際の入力、出力を条件に従ってレビューしてください



# テストと生成AI

## プロンプト

# 役割  
あなたは優秀なテスト設計者です  
# 前提条件  
テストケース策定に際してプログラムの仕様、データ制約、テストポリシーを考慮する  
# タスク  
プログラムの仕様を確認するためのテストケースを生成



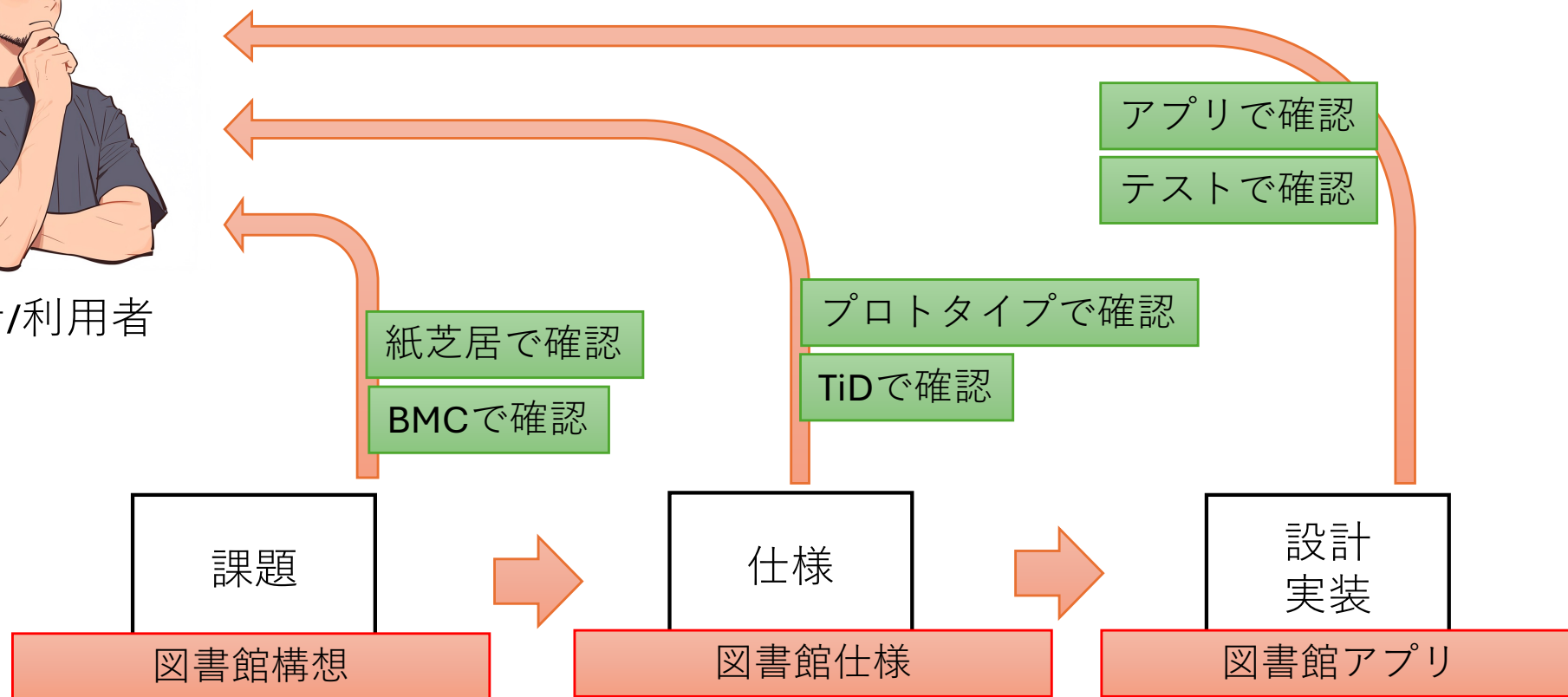
# 正しいものを作っているか？

## Validation

これまでは人間が目視で確認していたが ...



発注者/利用者

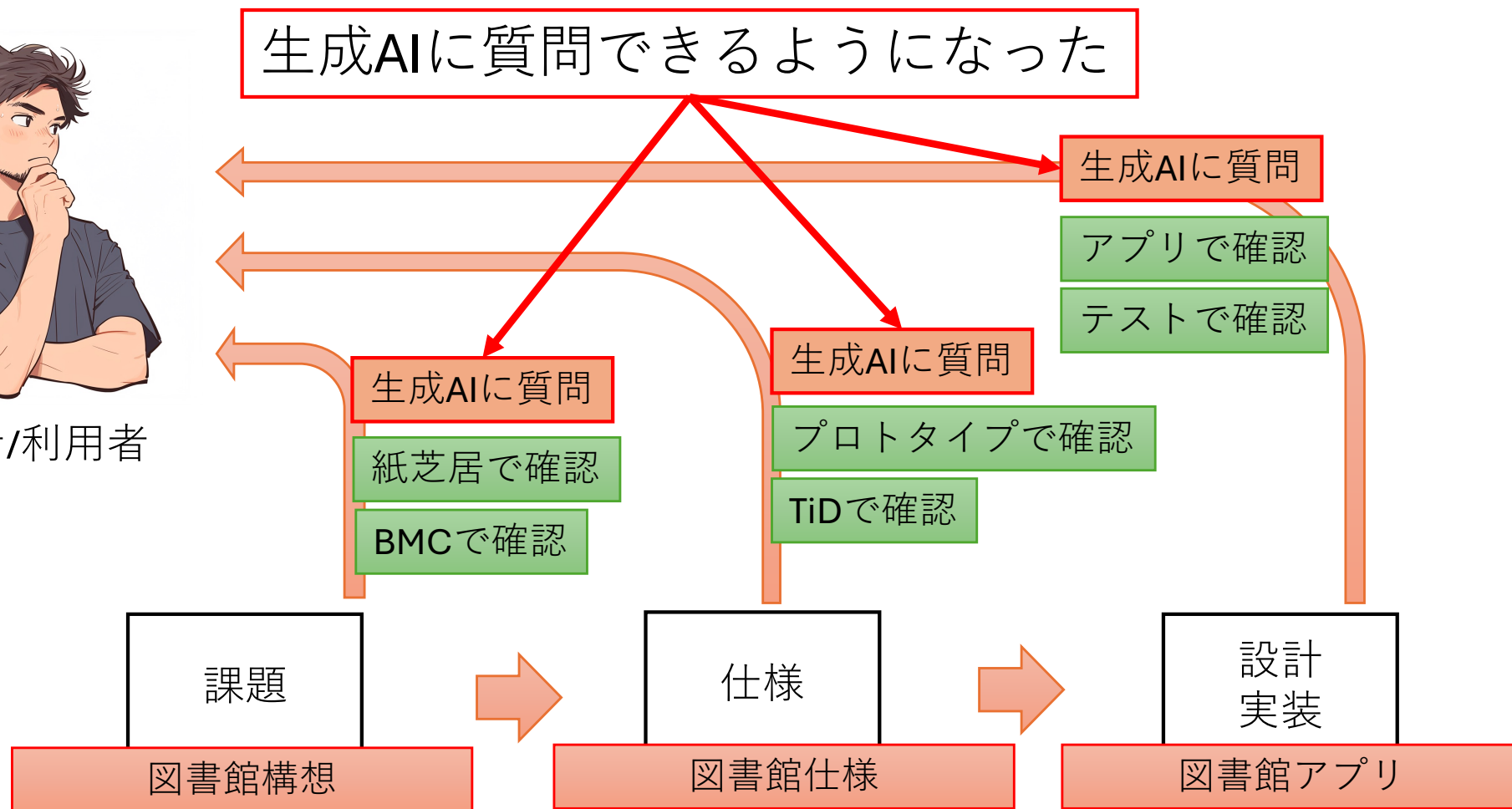


# 正しいものを作っているか？

## Validation



発注者/利用者



# 組織内知識共有

RAG（検索拡張生成）



# RAG（検索拡張生成）とは

- LLMが事前に持っていない個別の知識を取り込む
  - 社内規定
  - プロジェクトポリシー
  - 共通文書
  - Etc ...
- 三種のアプローチ
  - フレームワークとAPIを駆使したプログラミング
  - ノーコード、ローコード環境を使う手法
  - RAGを簡単に実現できるアプリケーションの利用

# RAG：フレームワークとAPIを駆使したプログラミング

- 生成AIを提供する各社のAPIを利用して、自分で使うAIを作る
  - AIコアになるエンジンそのものはなかなか作成できないが、たとえばメタ社の提供するLlama3などのオープンソースも増えている
  - オープンソース版を使えば、計算リソースは必要ですがオンプレミス、ローカルPC上の環境を作成することも可能
- AIコア以外の部分を作成して、動的に外部の情報を取り込むことでRAGを実現することができる
  - **LangChain** などのフレームワーク

# RAG：ノーコード、ローコード環境を使う手法

- LangChain のような仕組みを使って自作するのは敷居が高い
  - このためノーコード、ローコード的な手法を使って難しさを覆い隠し、比較的取り組みやすくした環境も出始めている
- 代表的なものは **Dify** という環境で、固有の知識をアップロードした上で、様々な生成AIのサービスをビジュアルにつなげて行き、プロンプトと組み合わせることで、**RAG**対応のチャットボットやワークフローを作成することが可能

# RAG : RAG を簡単に実現できるアプリケーションの利用

- Google の NotebookLM を使うと気軽に RAG 環境を用意できる



「伝えること」

と

「翻訳すること」

もう一つの視点

# 別の視点：何のための「翻訳」？

やりたいのは「**翻訳**」？それとも「**伝達**」？

もしやりたい事が「伝達」ならChatGPTに「翻訳」を頼むのではなく「伝えたい事」を教えて作文してもらおうという戦略も。

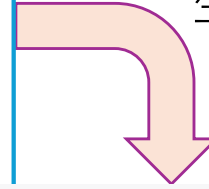
→ **Generative** の特性を最大に活かす方向性へ

以下の内容を伝える文章を英語で作って下さい。  
出力形式はマークダウン形式とします。  
多くの応募を促す文言も入れてください。

-----

青空キャンプ場予約開始  
受付は4月21日10時から  
応募資格は18歳以上なら誰でも  
ウェブページ ([aozora-camp.com](http://aozora-camp.com)) のみで受付  
第三希望まで入力して、応募書多数の場合は抽選  
お問い合わせはメールアドレス [xxx@yyy](mailto:xxx@yyy) まで

もちろん最終チェックは必要  
だが、日本語の原稿を「翻  
訳」するよりは自然な文章が  
生成される可能性が大きい？



## Aozora Campground Reservations Now Open!

Get ready for an unforgettable experience! Aozora Campground is excited to announce that **reservations will open on April 21st at 10 AM**. Don't miss this fantastic opportunity – we encourage everyone to apply and secure their spot!

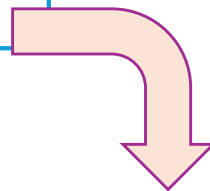
**Eligibility:** If you're 18 years old or older, you're welcome to join us!

**How to apply:** Reservations will only be accepted through our website: [aozora-camp.com](http://aozora-camp.com)

**Preference:** You can enter up to three preferences. In case of a large number of applications, spots will be allocated via a lottery system.

**Contact us:** If you have any questions or concerns, please feel free to reach out to us at our email address: [xxx@yyy](mailto:xxx@yyy)

同じ内容を中国語で作って下さい



## 青空露营地预订开始啦！

准备好享受一次难忘的体验吗？青空露营地激动地宣布，**预订将于4月21日上午10点开始**。不要错过这个绝佳的机会——我们鼓励大家申请并确保自己的席位！

**资格要求：**只要您年满18岁，就可以加入我们！

**如何申请：**预订只通过我们的网站接受：[aozora-camp.com](https://aozora-camp.com)

**选择偏好：**您可以填写最多三个选择。如果申请人数众多，席位将通过抽签分配。

**联系我们：**如果您有任何疑问或顾虑，请随时通过我们的电子邮件地址联系我们：[xxx@yyy](mailto:xxx@yyy)

我们迫不及待地欢迎您加入青空露营地！

逆翻訳をした限りではそれなりに正しそう ... ?



もちろん日本語でも ...

以下の内容を伝える文章を英語で作って下さい。  
出力形式はマークダウン形式とします。  
多くの応募を促す文言も入れてください。

-----

青空キャンプ場予約開始  
受付は4月21日10時から  
応募資格は18歳以上なら誰でも  
ウェブページ ([aozora-camp.com](http://aozora-camp.com)) のみで受付  
第三希望まで入力して、応募書多数の場合は  
お問い合わせはメールアドレス [xxx@yyy](mailto:xxx@yyy) まで

## 青空キャンプ場予約開始！🏕️大募集中！

みなさん、待ちに待った青空キャンプ場の予約受付が4月21日10時から開始されます！🎉  
あなたもこの機会に、忘れられないキャンプ体験をしませんか？多くの方々のご参加をお待ちしています！

### 応募資格

18歳以上であれば、どなたでもご応募いただけます。

### 予約方法

ウェブページ [aozora-camp.com](http://aozora-camp.com) にアクセスし、専用フォームからお申し込みください。受付はウェブページのみとなります。

### 第三希望まで入力

予約時には、第三希望までの希望日を入力してください。もし応募が多数の場合、抽選によって予約が決定されます。

### お問い合わせ

お問い合わせは、メールアドレス [xxx@yyy](mailto:xxx@yyy) までお願いいたします。

ぜひお早めにお申し込みください！大自然の中で素晴らしいキャンプ体験が待っています！🌲🔥

## 他の例：沖縄の医師である高山義浩氏の投稿より

- 日本の公立病院が提供する食事が口に合わない外国人患者のため、ウーバーイーツを注文することを容認する英語の説明文書を作ってください。文書に盛り込むべき点は以下です。
- 注文できるのは昼食のみです。食中毒のリスクがあるので、生ものはダメです。お菓子類は注文しないでください。
- 注文するときは、必ず看護師に事前に言ってください。何を注文するのも口頭で教えてください。
- 注文はクレジット決済としてください。病院は現金による支払いを代行しません。
- 入院中に体調の変化があるとき、検査を予定しているときなど、注文しないようにお願いすることがあります。従ってください。

### Title: Ordering Uber Eats for Foreign Patients at Japanese Public Hospitals

In response to foreign patients who may find the meals provided by Japanese public hospitals not to their liking, we have decided to allow ordering from Uber Eats under certain conditions. Please take note of the following points when placing an order:

You may only order lunch. To minimize the risk of food poisoning, please avoid ordering any raw food items. Additionally, please refrain from ordering sweets and snacks.

Before placing your order, please inform the nurse in charge. Verbally communicate your intended order to the nurse as well.

Please use a credit card for payment. The hospital will not handle cash payments on your behalf.

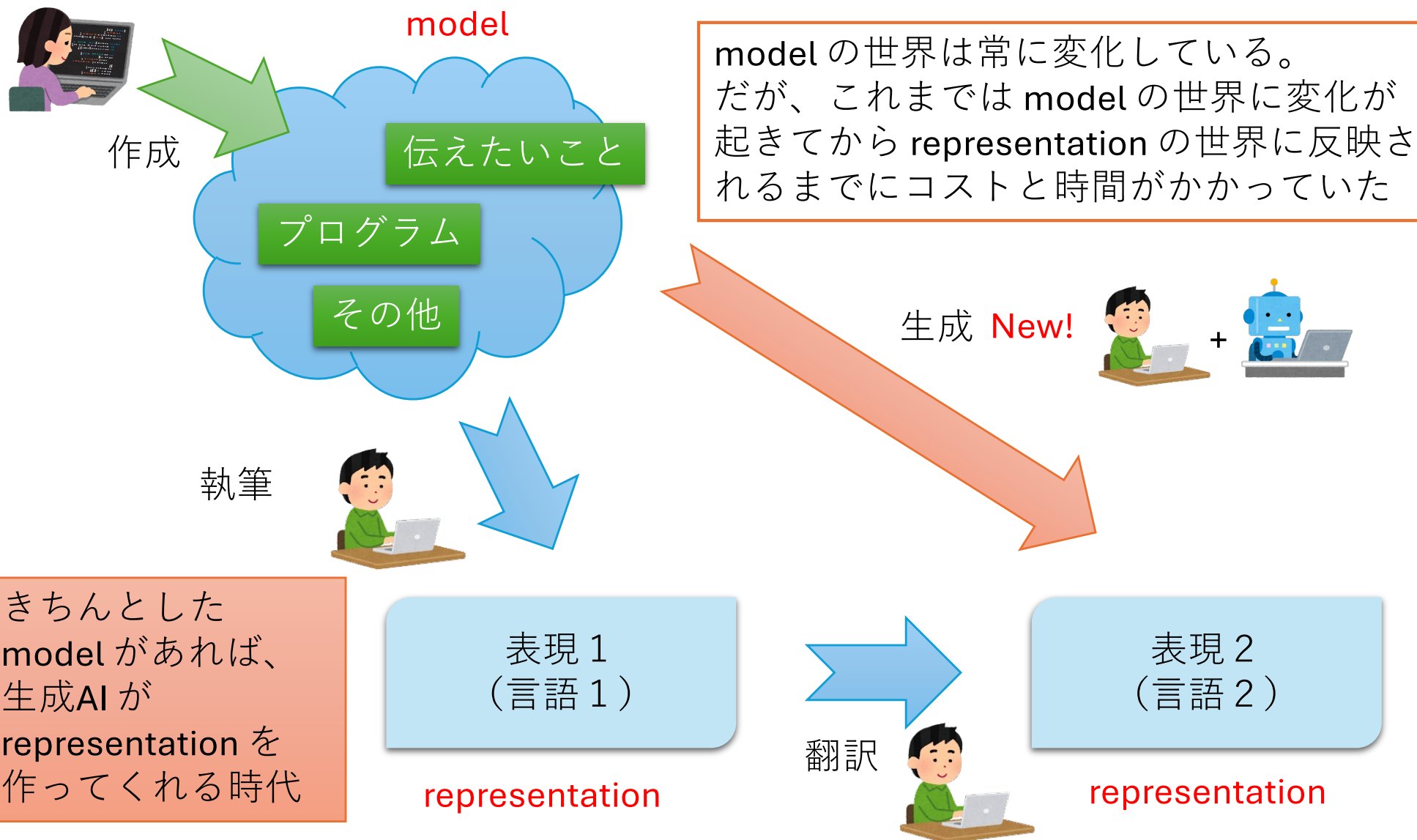
There may be instances when we ask you not to place an order, such as when you experience changes in your physical condition during your hospital stay or when tests are scheduled. In these cases, please comply with our request.

翻訳そのものではなく、伝達内容に基いた成果物

# 翻訳 vs 生成

- 「翻訳」は何も英語から日本語への変換だけを指しているわけではない
- ある記述を別の記述へ、目的に応じて書き換える行為も「翻訳」の一種と考えて良い
- たとえば
  - 要約もパラフレーズも翻訳の一種
  - 箇条書きを文章にするのも翻訳の一種
  - 常識で行間を埋めるのも翻訳の一種 ...
  - Etc ...
- これらの作業に **Generative AI** は大きく関わってくる

# Model から Representation へ



プロンプト  
エンジニアリング

# プロンプトを書く際の 2 原則

## • 原則 1

- 明瞭で具体的な指示を書くこと
  - 何が欲しいのかをはっきりさせる

## • 原則 2

- モデルに「考える」余裕を与えること
  - 複数のことをいっぺんに考えさせると混乱する

# (現在) 有名な手法

- ゼロショット (Zero-shot)
- フューショット (Few-shot)
- チェインオブソート (Chain of thought)
- プロンプトプライミング (Prompt priming)
- インストラクション (Instruction)
- アンカリング (Anchoring)
- ネガティブプロンプト (Negative Prompt)
- ... etc ...

細かいテクニックはいろいろあるが一所懸命覚えてもあまり意味はない

なぜなら... (次のスライドへ)

プロンプトエンジニアリングの現在

# ほぼ無意味

自分でまずどのように問題を解決すべきかを推論するモデルの登場

ChatGPT の o1, o3 モデル

DeepSeek の R1モデル

プロンプトを作成するツールの存在

OpenAI の playground

<https://platform.openai.com/playground>



# プロンプトエンジニアリングの現在

なので何十万円もかかる  
「プロンプト講座」  
(名称は様々) は ... ?

ChatGPT の o1, o3 モデル

DeepSeek の R1モデル

プロンプトを作成するツールの存在

OpenAI の playground

<https://platform.openai.com/playground>

ということで各論は忘れて

## プロンプトエンジニアリングの基礎（1）

### プロンプト構成要素

- 文脈（ロール）の設定
- 行ってほしい内容（目的、制約、手段）
- 出力形式
- 詳細化への促し

## 各論以前の プロンプトエンジニアリングの基礎（2）

あなたは要件定義アシスタントです。相手の書く文章を解析して、アクター、アクターが最終的に行うこと、そのために満たされているべき前提条件に分解し、それぞれを箇条書きにしてください。もし不明な点ある場合は何でも質問してください。

### プロンプトの工夫の基本

- (1) AIにロールを割り当てる
- (2) 行って欲しいこと
- (3) 出力形式
- (4) 詳細化への促し

### 文章

「唐津市は景気浮揚策として商品券を販売しようとしています。販売対象は市民です。販売される商品券は額面5000円、販売価格4000円で、プレミアが1000円分乗せられています。商品券の販売数には限りがあり上限10000枚が予定されています。人気が予想されるため、事前申請と抽選を行い当選者に商品券を販売することにします。事前申請で申し込めるのは最大4枚までです。この業務を行うために必要な手続きを挙げて下さい」

# 唐津市：実行例（1）

要求に従った出力が得られている

分解：

## アクター：

- 唐津市
- 市民

## アクターが最終的に行うこと：

- 商品券の販売（唐津市）
- 商品券の購入（市民）
- 満たされているべき前提条件：
- 販売される商品券は額面5000円、販売価格4000円
- 商品券のプレミアは1000円分
- 商品券の販売上限は10000枚
- 事前申請と抽選を行う
- 当選者に商品券を販売する
- 事前申請で申し込めるのは最大4枚まで

## この業務を行うために必要な手続き：

- 商品券の販売計画を立案（唐津市）
- 事前申請方法と抽選方法を決定（唐津市）
- 事前申請期間を設定（唐津市）
- 市民に事前申請方法と抽選方法を周知（唐津市）
- 市民が事前申請を行う（市民）
- 抽選を実施（唐津市）
- 当選者に連絡（唐津市）
- 商品券の販売と配布（唐津市）
- 市民が商品券を利用（市民）
- 不明な点：
- 事前申請の方法は何ですか？
- 抽選方法にはどのような基準がありますか？

## 不明な点：

- 事前申請の方法は何ですか？
- 抽選方法にはどのような基準がありますか？

質問が出された

# 唐津市：実行例（2）

質問への回答

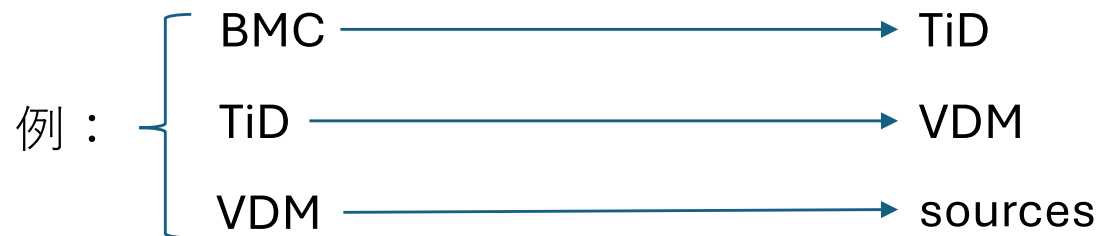
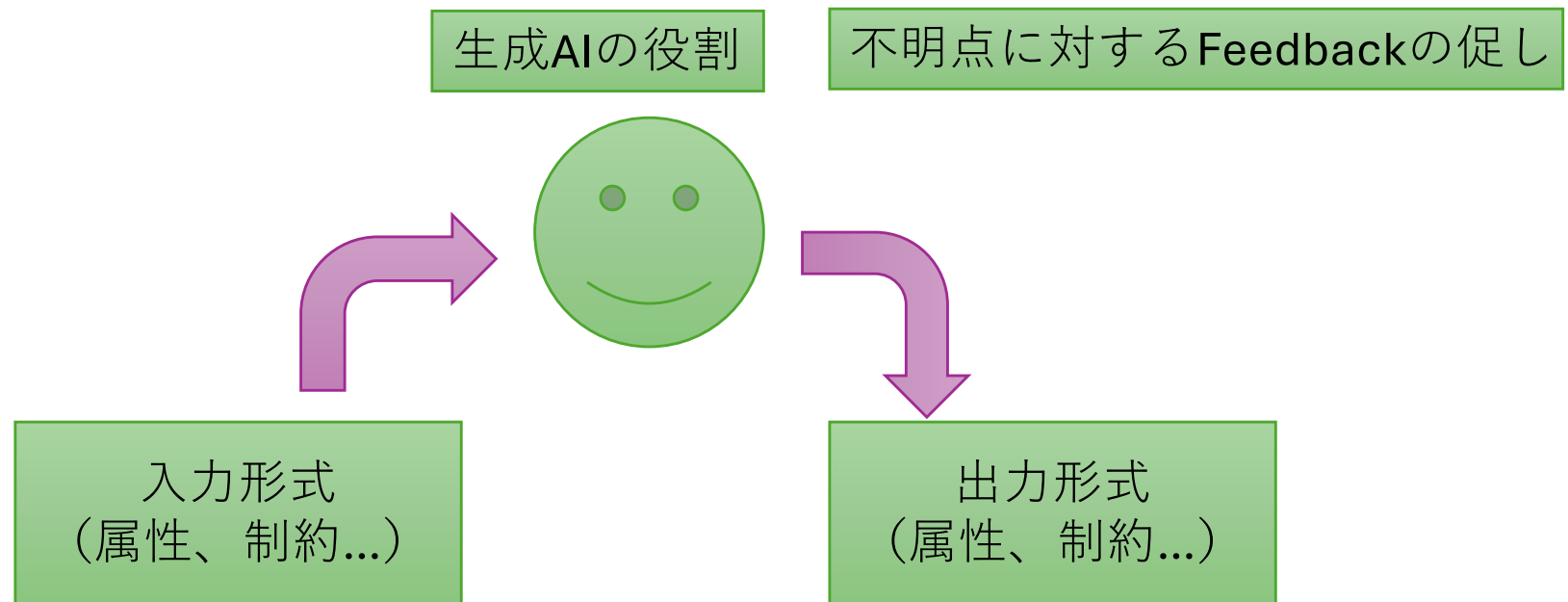
【人間】事前申請はウェブを使った申込みとします。申込時に市民は自分のアカウントを作成します。抽選方法は申請からランダムに申込商品券数が10000枚になるまで選び続けます。

- **アクター:**
  1. 唐津市
  2. 市民
- **アクターが最終的に行うこと:**
  1. 商品券の販売（唐津市）
  2. 商品券の購入（市民）
- **満たされているべき前提条件:**
  1. 販売される商品券は額面5000円、販売価格4000円
  2. 商品券のプレミアは1000円分
  3. 商品券の販売上限は10000枚
  4. 事前申請と抽選を行う
  5. 当選者に商品券を販売する
  6. 事前申請で申し込めるのは最大4枚まで
  7. 事前申請はウェブを使った申込み
  8. 申込時に市民は自分のアカウントを作成
  9. 抽選方法は申請からランダムに申込商品券数が10000枚になるまで選び続ける

質問に答えた結果

- **この業務を行うために必要な手続き:**
  1. 商品券の販売計画を立案（唐津市）
  2. 事前申請方法と抽選方法を決定（唐津市）
  3. 事前申請期間を設定（唐津市）
  4. ウェブ申請システムを開発・準備（唐津市）
  5. 市民に事前申請方法と抽選方法を周知（唐津市）
  6. 市民がウェブ上でアカウントを作成（市民）
  7. 市民がウェブ上で事前申請を行う（市民）
  8. 抽選を実施（唐津市）
  9. 当選者に連絡（唐津市）
  10. 商品券の販売と配布（唐津市）
  11. 市民が商品券を利用（市民）

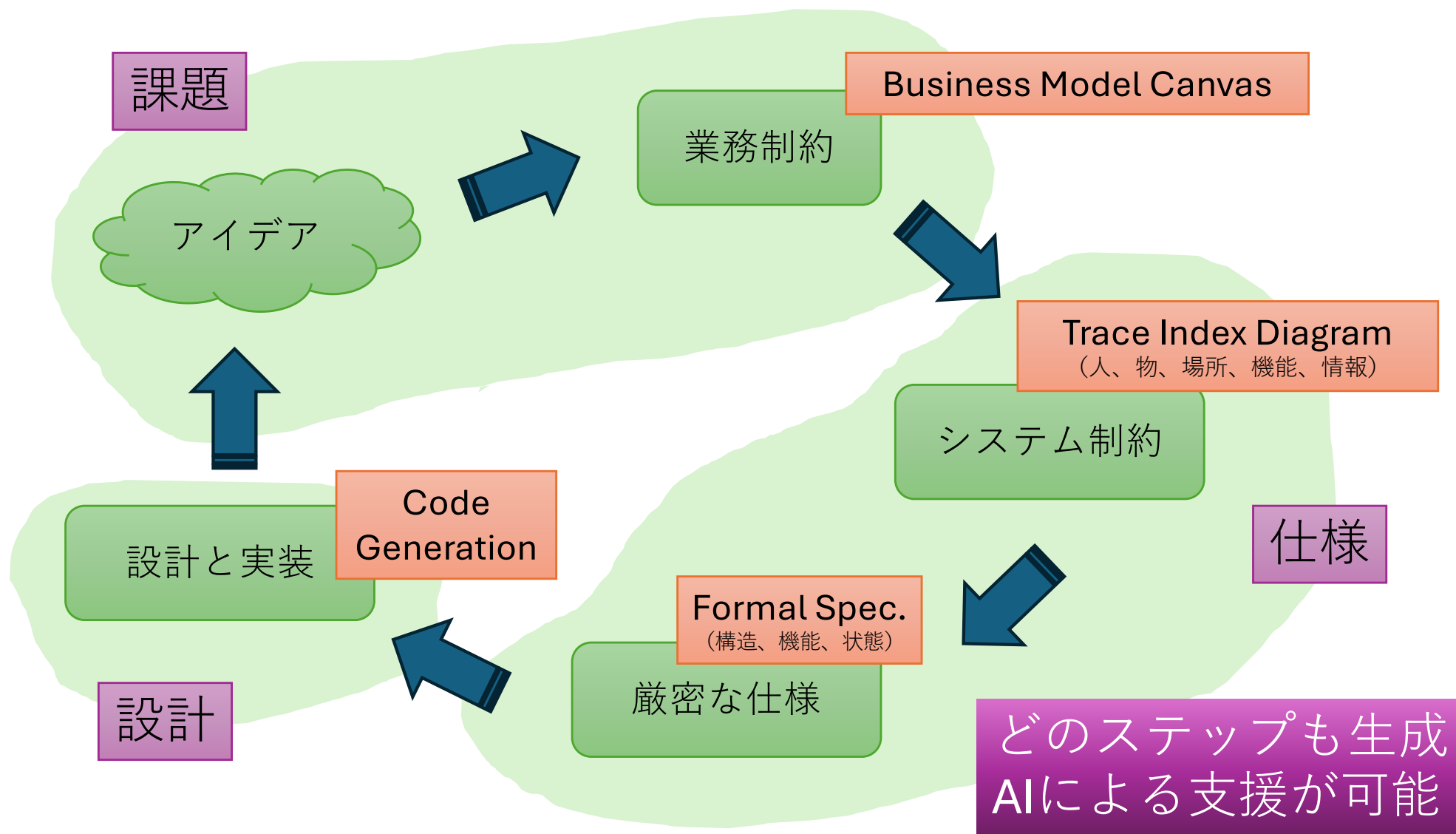
# プロンプトに与えたいメタ情報



etc ...

まとめ (?)

# 振り返り：開発サイクルと生成AI



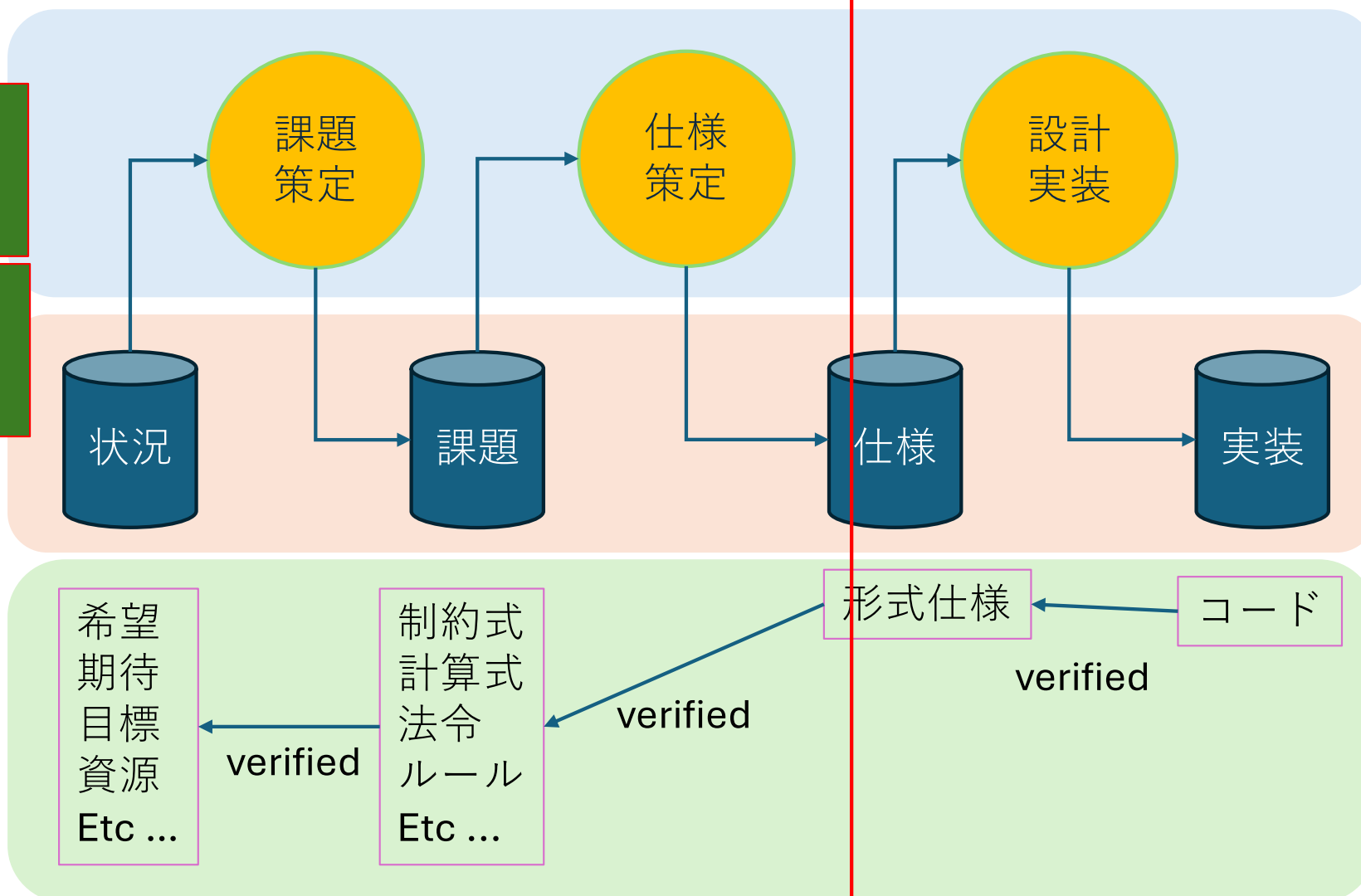


# 生成AIの役割

No Code/Low Code

生成AIは全ての  
プロセスを  
支援

どの段階でも  
Validation を可  
能にする



# 生成AIのメリットとデメリット

- 長所

- 豊富なバリエーションの中から検討できる
- クリエイティブな提案ができる
- 上手く使えばコストの大幅な削減につながる
- レビューの観点などを揃えることができる

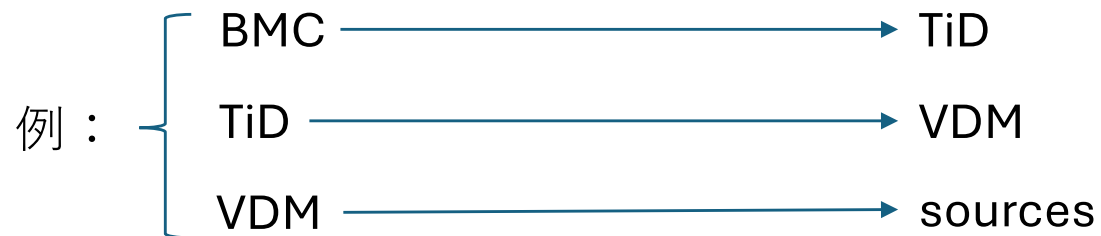
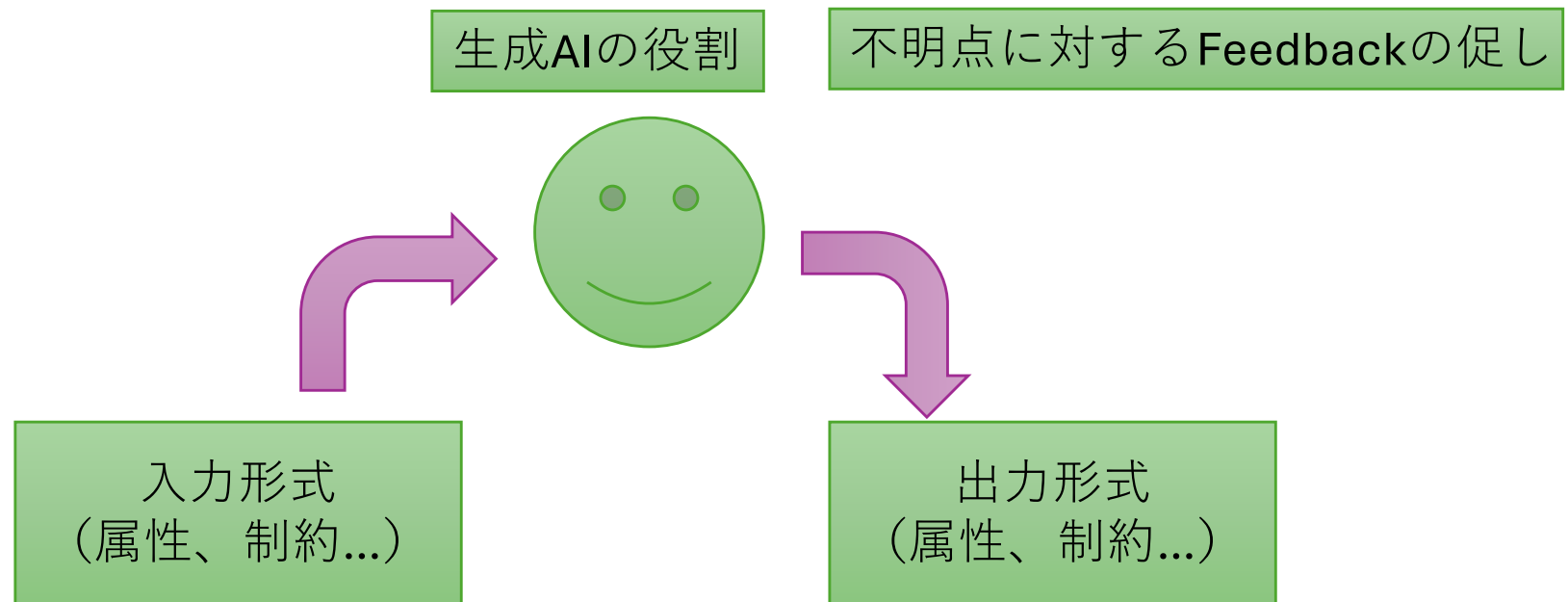
- 注意点

- 生成された結果の品質や信頼性の担保は人間の責任
- 人間の感性や価値観を置き換えることはできない

# 生成AIの効果的な利用のために

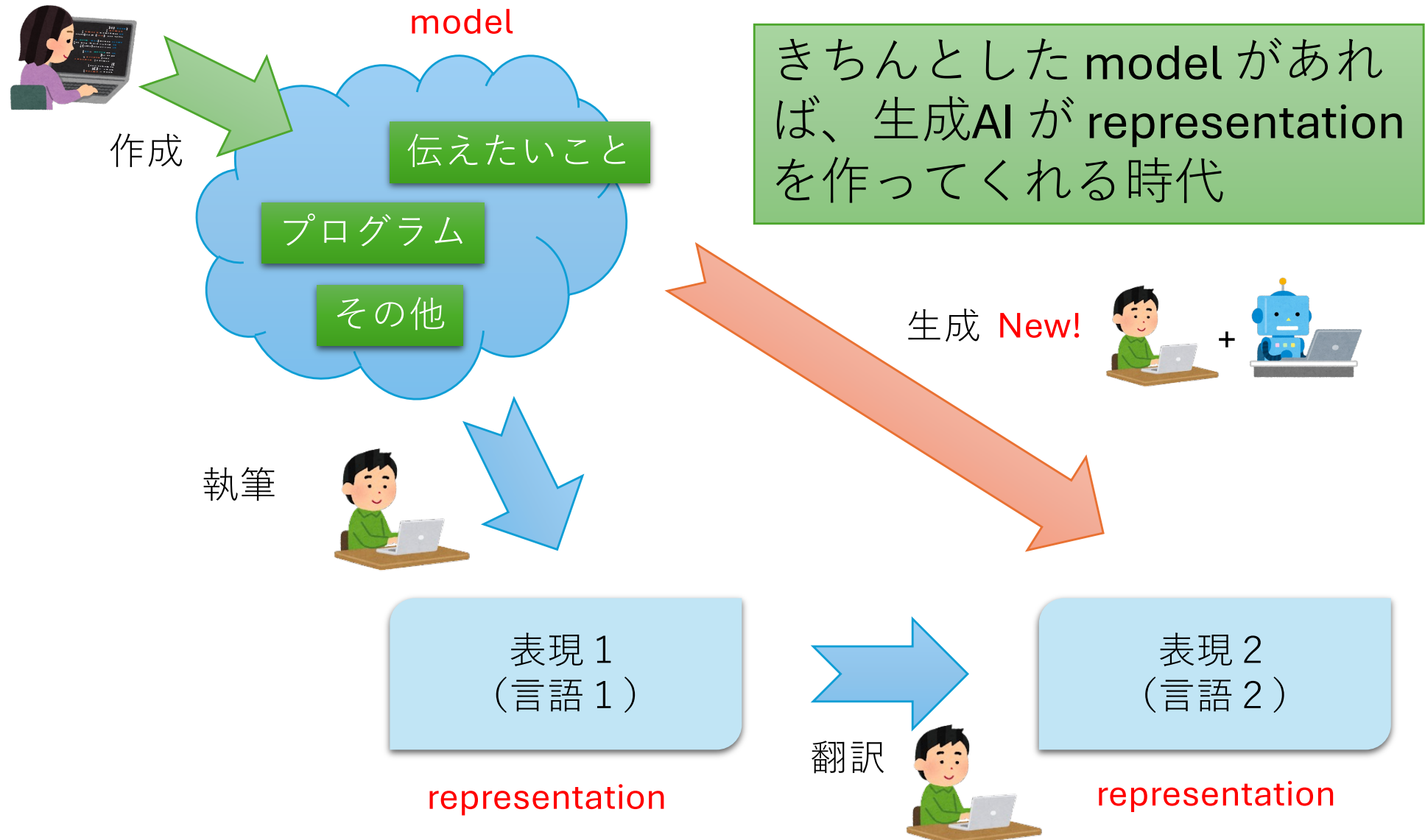
- 漠然とした質問には漠然とした答しか返ってこない
  - 徐々にでも良いので欲しいものの「制約条件」(What)をはっきりさせていくこと
- きちんとした成果物があるならそれを前提に、欲しい解析結果を指定して依頼すると効果的
  - 厳密な仕様記述（モデル）は生成AIを使う際にも大きな助けになる
- 要求定義、仕様策定、設計、テストケース生成、テストデータ生成などにふさわしいプロンプトを工夫してライブラリ化
  - それぞれのステップで欲しいモデルの性質の明確化

# プロンプトに与えたいメタ情報

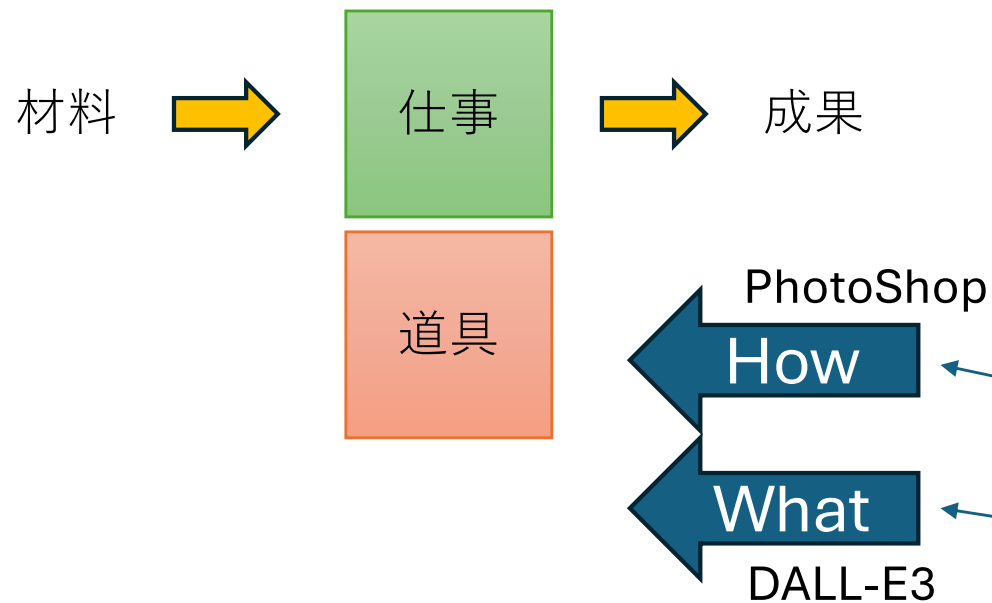


etc ...

# Model から Representation へ



# では「道具」とは？



- これまでの道具は **How** を与えて微調整しながら成果を得るものだった
- だが生成 AI によって作られる道具は **What** を与えて成果を得るものとなる

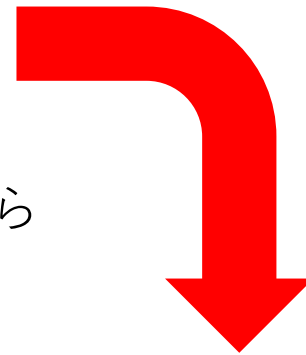


最後に ... 生成 AI が後押しするのは

from

How

の時代から



to

What

の時代への動き

ゴールを描ければたどり着く方法は教えてもらえる  
(かも)

# 蛇足：洞察の階層

