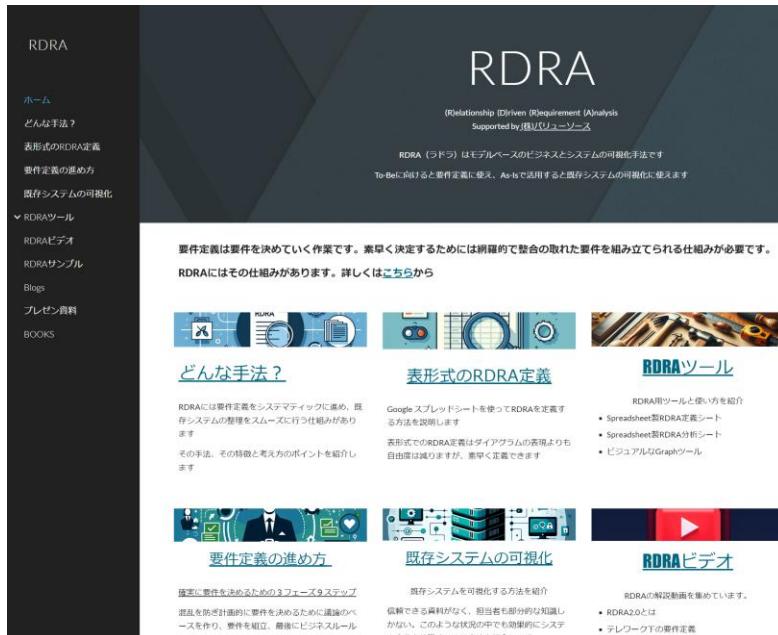


CursorでAI活用のナレッジ  
ベースを構築する

- (株)バリューソース
  - 神崎 善司
  - twitter : @zenzengood
  - RDRA : <https://www.rdra.jp/>
- 著作
  - モデルベース要件定義テクニック
  - RDRA2.0ハンドブック
  - RDRA3.0ハンドブック
- 仕事
  - RDRA導入支援
  - 要件定義支援
  - 既存システムの可視化支援
- 好きな事
  - システムの可視化
  - モデリング
  - 表形式でのモデリング
  - LLMの活用

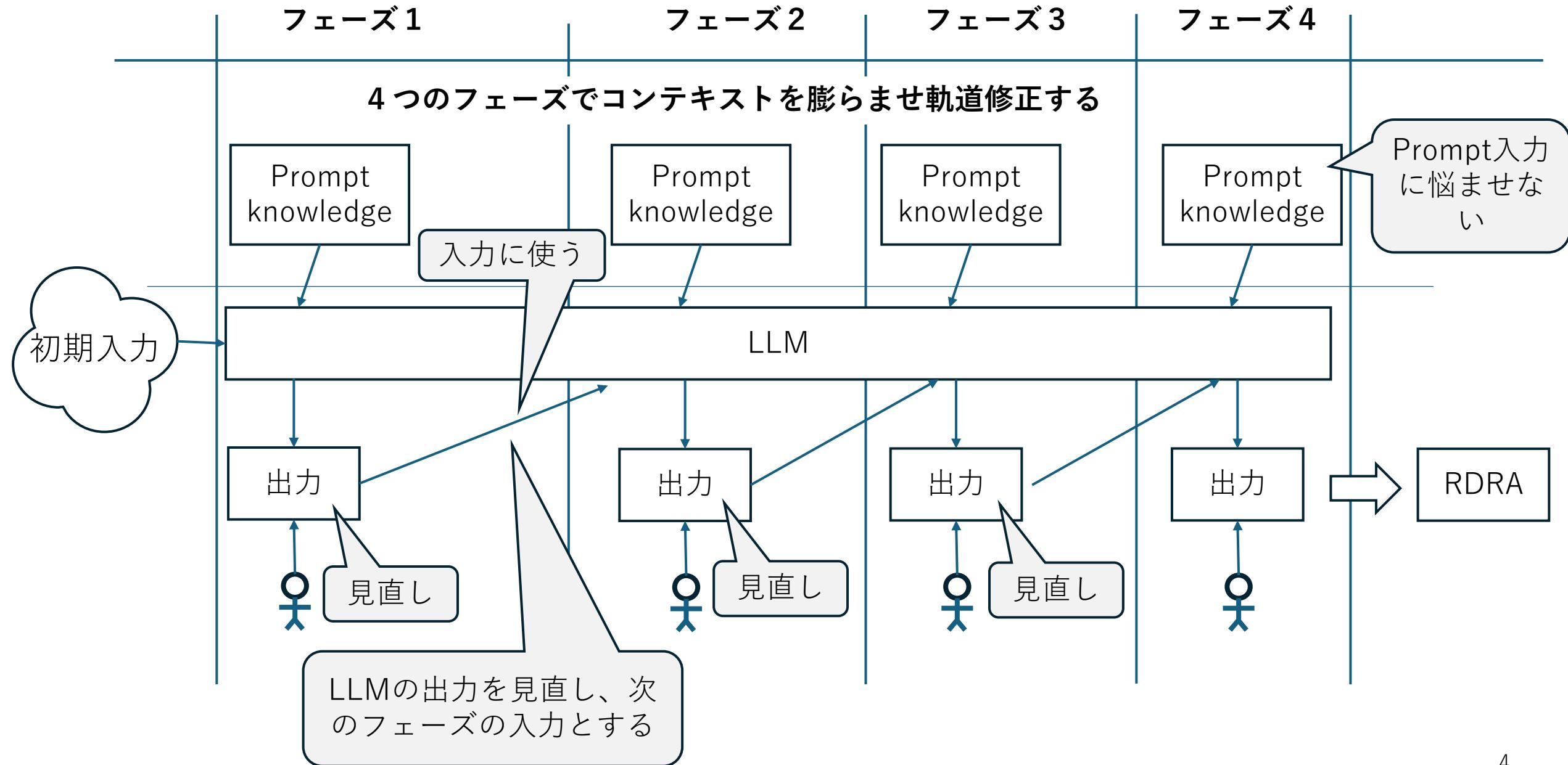
<https://www.rdra.jp/>



# LLMを使うまでの課題感

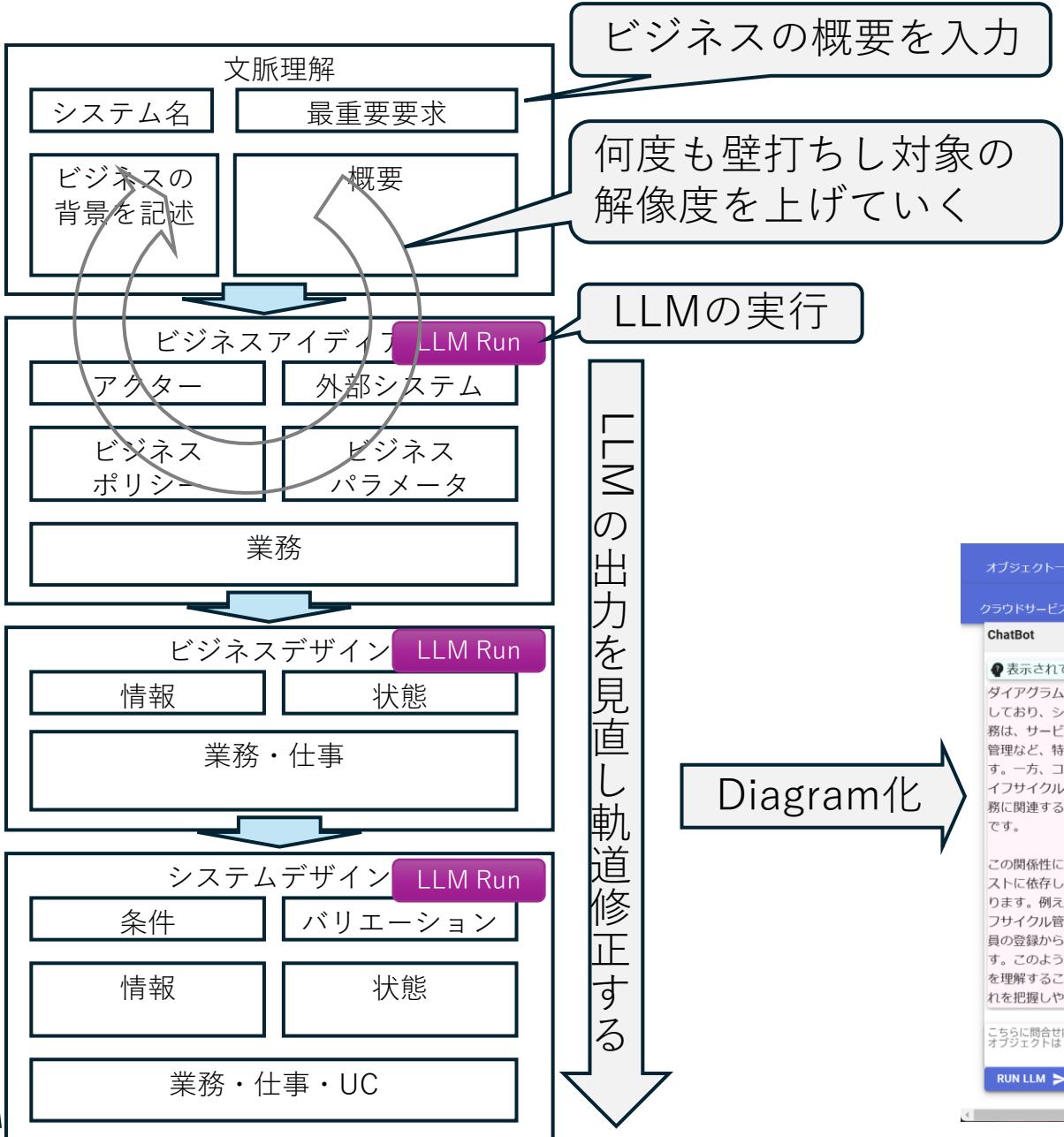
- エンタープライズ向けの要件定義
  - LLMには会社固有の情報は一切ない
- 課題
  - どうやって適切なコンテキストを用意するか
  - LLMの出力をどのように軌道修正するか
  - LLMの出力をいかに理解するか

# 多段に人とLLMがキャッチボール



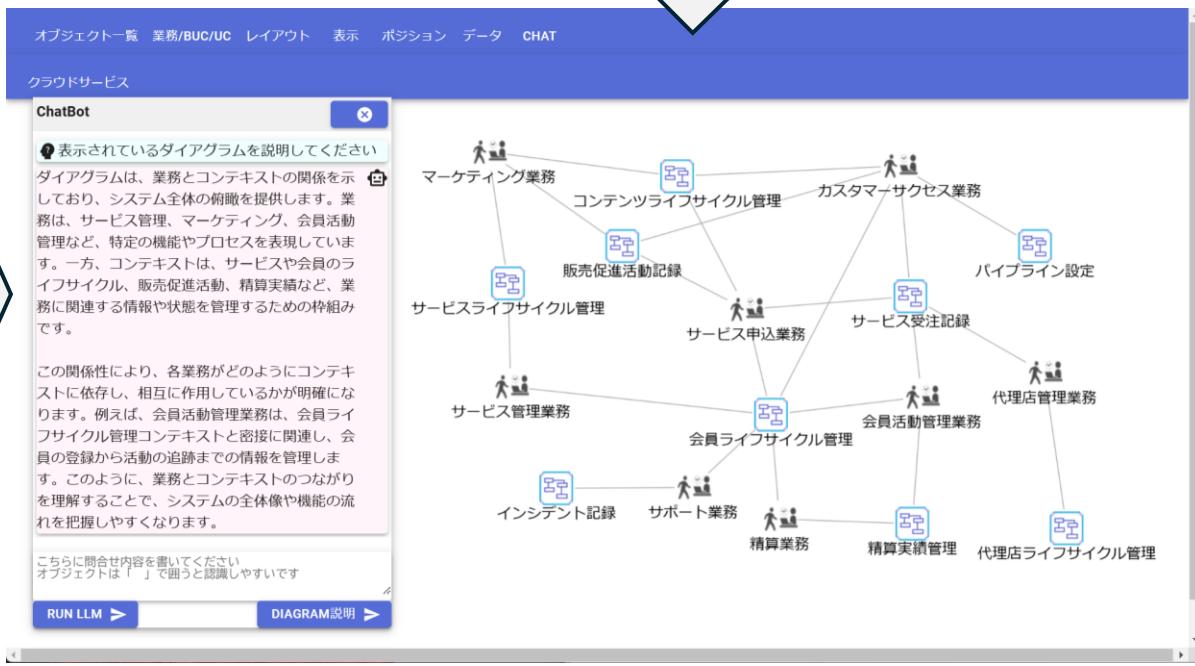
# LLMと人がキャッチボールを行い要件を組立てる

段階的にコンテキストを用意



- どうやって適切なコンテキストを用意するか
- LLMの出力をどのように軌道修正するか
- LLMの出力をいかに理解するか

RDRA定義を俯瞰して理解しやすくする



# LLMでRDRAを定義する

<https://vsco.jp/rdratool/v0.93/index.html>

↓文脈理解 ↓ビジネス\_アイディア ↓ビジネス\_デザイン ↓システム化検討 | LOAD SAVE CLEAR | TO RDRA TO GRAPH | SAMPLE | REGISTER API

文脈理解(Contextual understanding) : システム化対象をLLMに伝えるためにビジネスの背景や業務の概要を記述し初期情報を提供する

システム名 介護システム

最重要要求

多様な介護会員の状態に対応と、サービススタッフの多様な働き方を効率的計画で支援する

## ビジネスの背景

### ## 背景

介護業界では、介護会員の要望に合わせ、様々な法規の中でサービススタッフの多様な働き方でスケジュールを調整する必要がある

### ## 要求

・サービススタッフの柔軟な働き方に対応しあつ介護会員の要望に合わせたスケジュール調整を行える仕組みを持つ

### ## ビジネスピリシー

- 各サービススタッフの要望に合わせたスケジュール管理を行う
- 訪問介護事業で小規模事業社を複数持つので、そこを一元管理を行う事業所をもつ
- サービススタッフの働き方は出来るだけ自由に設定できる
- サービススタッフのスキル別に出来る仕事を適切に配分する
- 介護会員の要望に合わせたスケジュール調整を行う
- ビジネスパラメータの組合せに柔軟に対応できる業務形態にする

### ## ビジネスパラメータ

- 介護会員とサービススタッフの自由度を保つために介護施設を分類できる
- サービススタッフの自由度を保つために働き方を細かく分類する
- 介護会員の状況を細かく分類できる

## 概要 (システム化対象で行っている事を箇条書きで記述)

- 介護会員の管理（名前、連絡先などの様々な属性を管理）
- 効率的な訪問介護の計画
- 訪問介護の実施記録を管理する
- 新しいサービススタッフの教育
- 介護費用の計算と請求及び回収

ビジネス アイディア(Business ideas) : システム化対象のアクター、外部システム、業務を一旦決めて次に進む

RUN LLM ▶

## アクター : ビジネスの背景内のアクターに基づき生成

I / M	アクター	役割	D
■	□	□	□
■	□	□	□
■	□	□	□
■	□	□	□
■	□	□	□

## 外部システム : ビジネスの背景内の外部システムに基づき生成

I / M	外部システム	役割	D
■	□	□	□
■	□	□	□
■	□	□	□
■	□	□	□
■	□	□	□

## ビジネスポリシー : ビジネスの背景に基づき生成

I / M	ポリシー	説明	D
■	□	□	□

## ビジネスパラメータ : ビジネスの背景に基づき生成

I / M	パラメータ	説明	D
■	□	□	□

# 見えてきた新たな課題



出力したものをどのように見直す？  
何に着目すればいいのか？

LLM・Cursorの性能は日進月歩で進化する

プログラムで作りこむと、柔軟性が失われる

- Promptを実行  
Tableで修正
- 繰り返す

RDRAZeroOne

Prompt

Table

JSON

LLM

必要な要素は全てIDEの中にある

Cursor

Prompt  
Text

Table  
tsv/addin

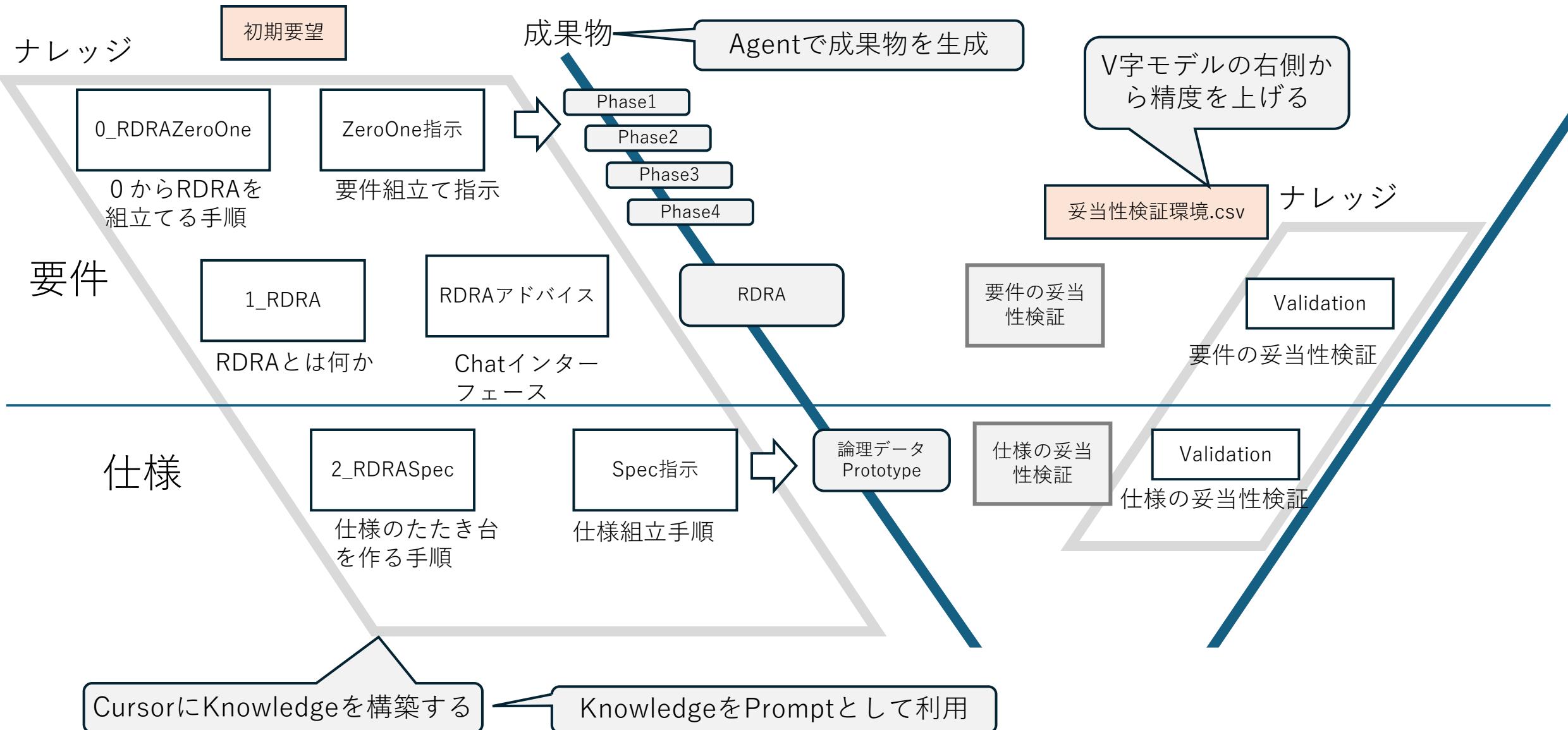
Agent

そうだ！  
Agentに要件定義させればいい

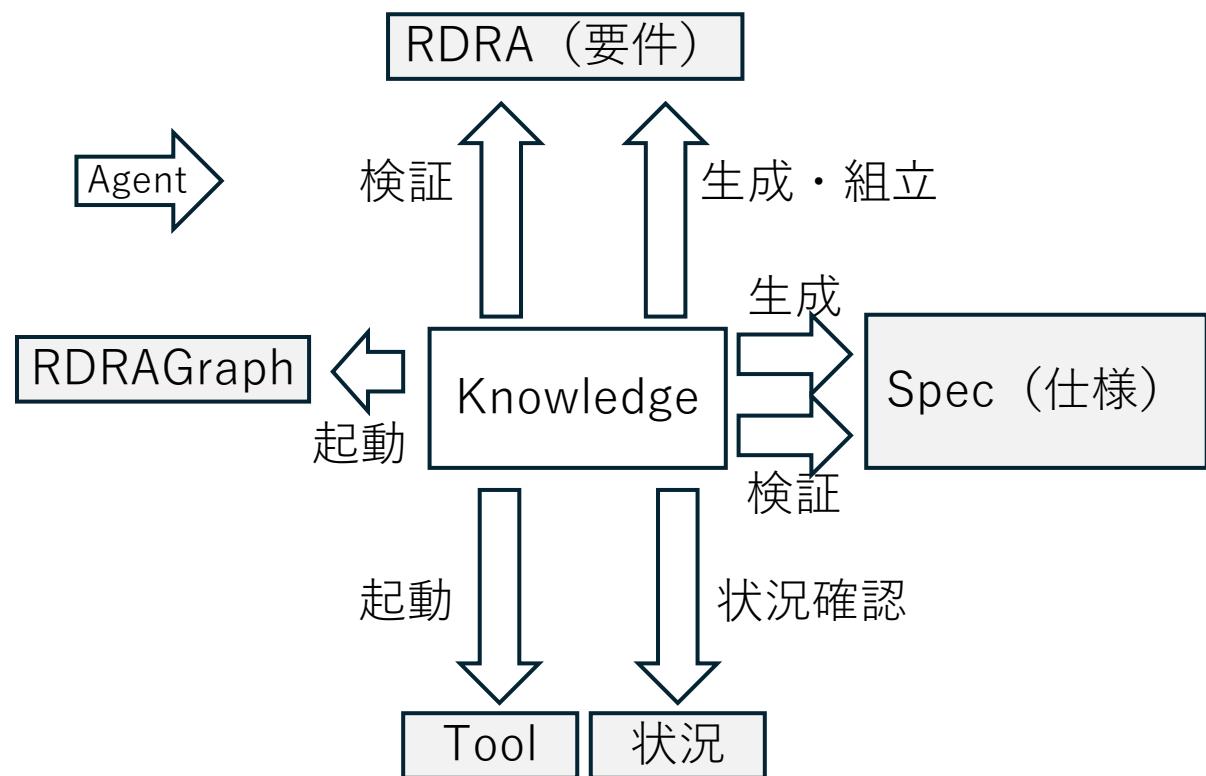
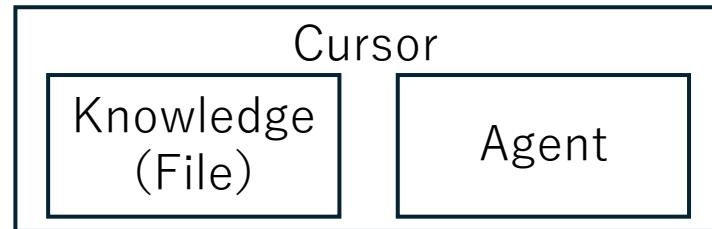
Cursor Agentを活用する

# 今回の環境

V字モデルを左と右から同時に進める

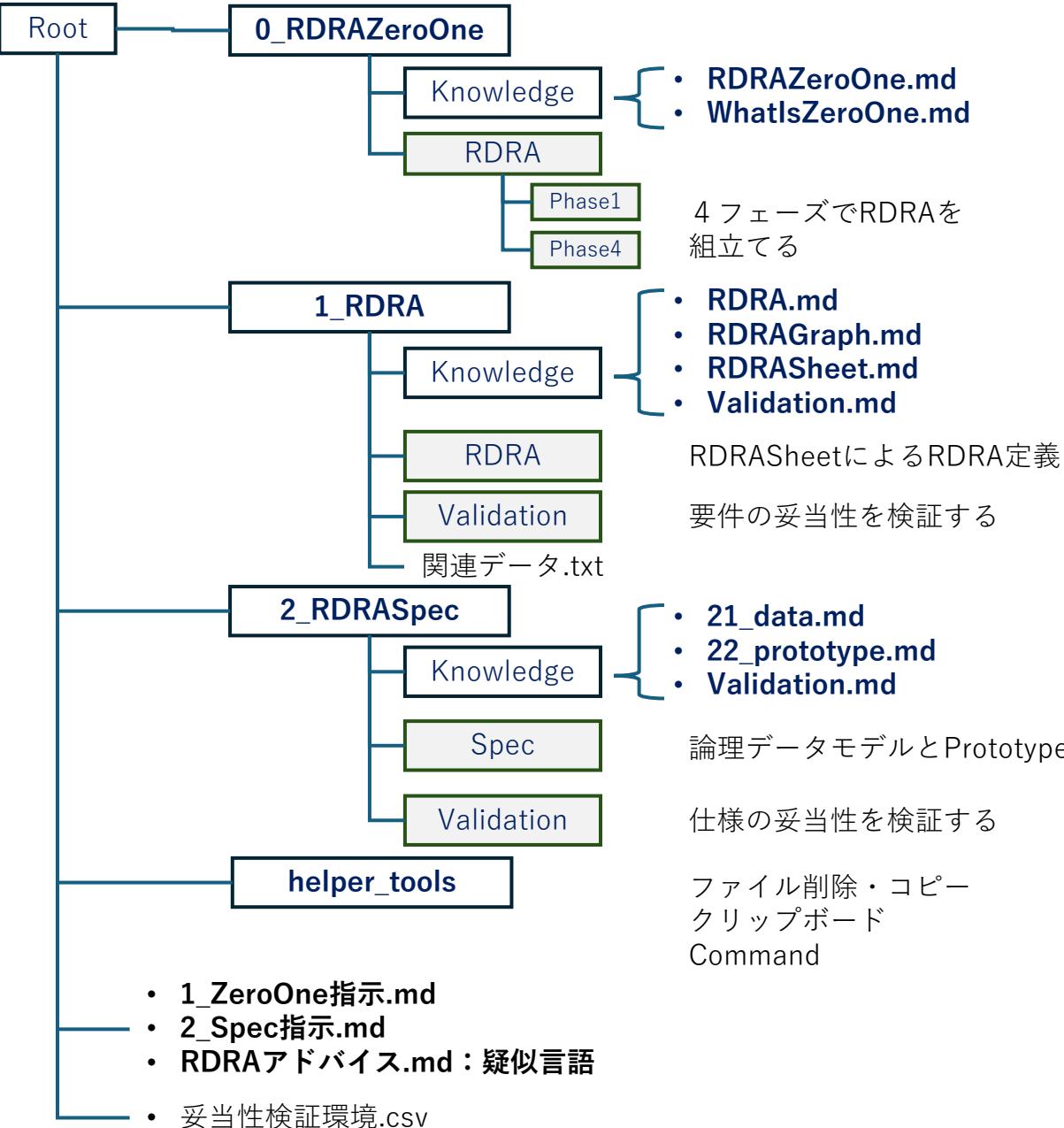


# どうのような環境にしたか



- 課題
  - AIモデルが日々進化する
  - Cursorが毎週アップデートする
- Agent
  - 要件 (RDRA) を組立てる
  - 状況を確認する : **疑似言語**を使う
  - RDRAGraphを表示する
  - 要件の妥当性を検証する
  - 仕様のたたき台を作成する
  - 仕様の妥当性を検証する
  - Toolを使う
- プロジェクト環境
  - 環境依存させない (一部環境依存)
  - テキストとJavaScriptだけで実現する

# プロジェクトの構成



Agentが生成するファイル

- 0\_RDRAZeroOne
  - 初期要望からRDRA定義を生成する
- 1\_RDRA
  - 生成したRDRA定義
  - RDRA定義の妥当性を検証する
- 2\_RDRASpec
  - 仕様を生成する
  - 仕様の妥当性を検証する
- helper\_tools
  - 単機能のツール ファイルコピー～
- 状況把握と指示ファイル
  - 1\_ZeroOne指示.md
  - 2\_Spec指示.md
  - RDRAアドバイス.md

# メニューで指示する

- メニューの表示
- 「…」／Clear All Chats
- ⇒ 「？」

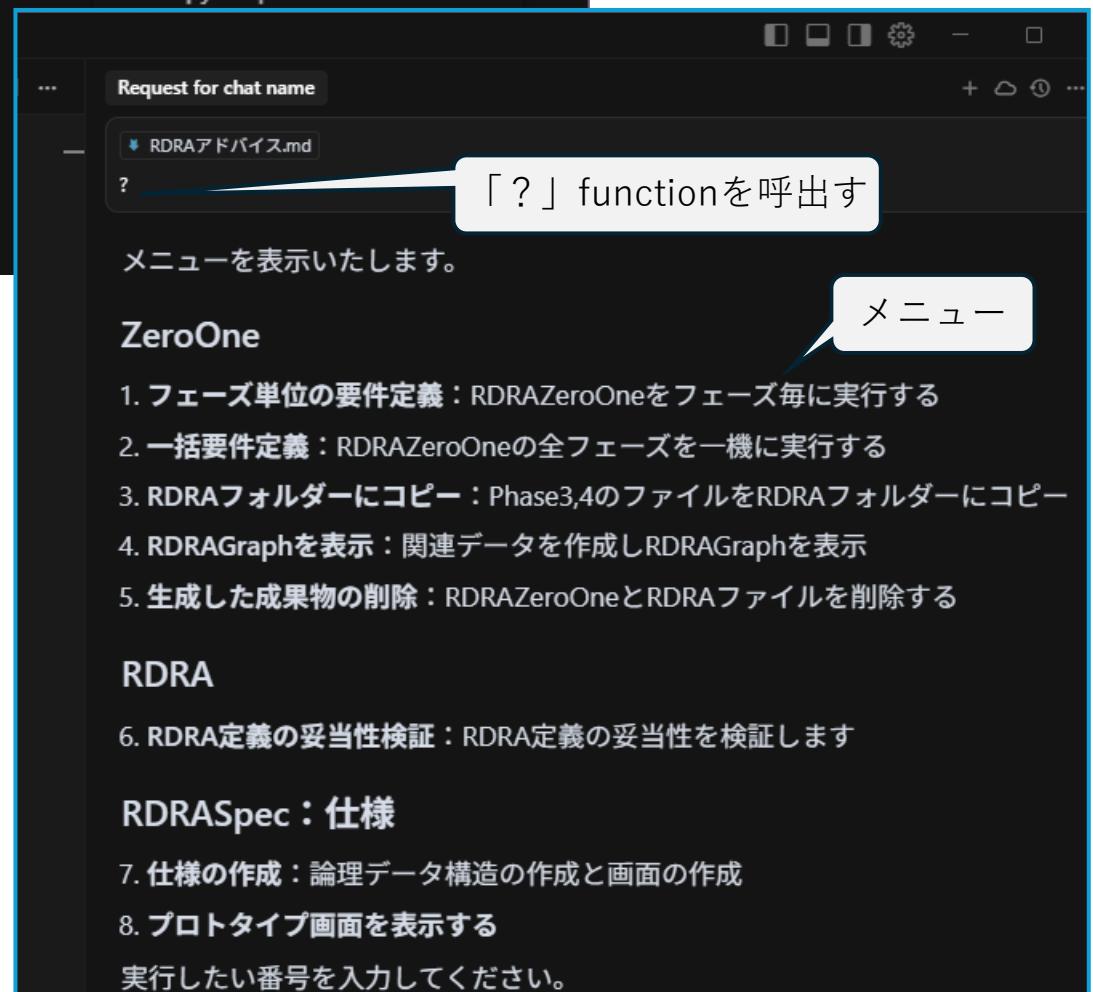
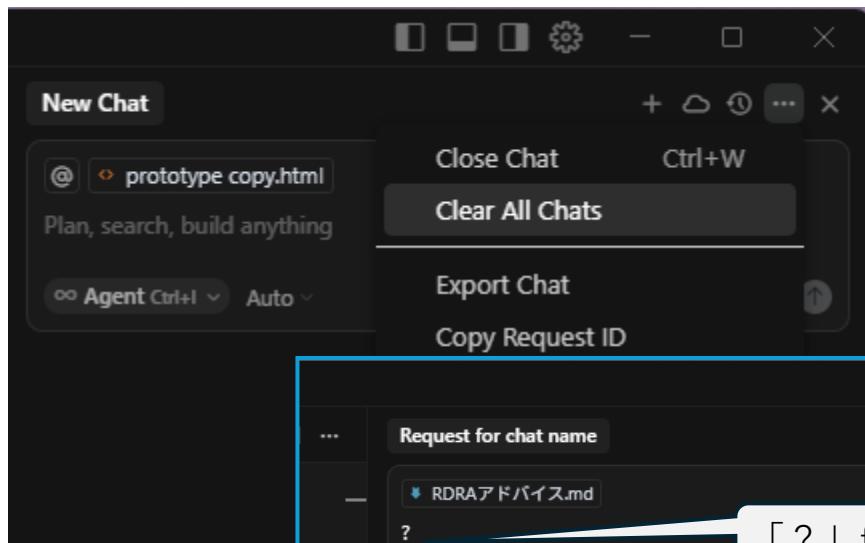
副作用を避けるために  
Chat履歴を全て消す  
「…」／Clear All Chats

RDRAアドバンス.md

```
function ?(){
    次のメニューを表示する
    ■ZeroOne
        1. フェーズ単位の要件定義：RDRAZeroOneをフェーズ毎に実行する
        2. 一括要件定義：RDRAZeroOneの全フェーズを一機に実行する
        3. 生成した成果物の削除：RDRAZeroOneとRDRAファイルを削除する
    ■RDRA
        4. 生成したファイルをRDRAフォルダーにコピーする
        5. RDRAGraphを表示：関連データを作成しRDRAGraphを表示
        6. RDRA定義の妥当性検証：RDRA定義の妥当性を検証します
    ■RDRASpec：仕様
        7. 仕様の作成：論理データ構造の作成と画面の作成
        8. プロトタイプ画面を表示する
        9. 仕様の妥当性を検証する
    実行した番号を入力してください
}
```

疑似言語

疑似言語でメニューを表示



# AgentでRDRAを定義する

Phase1～Phase4

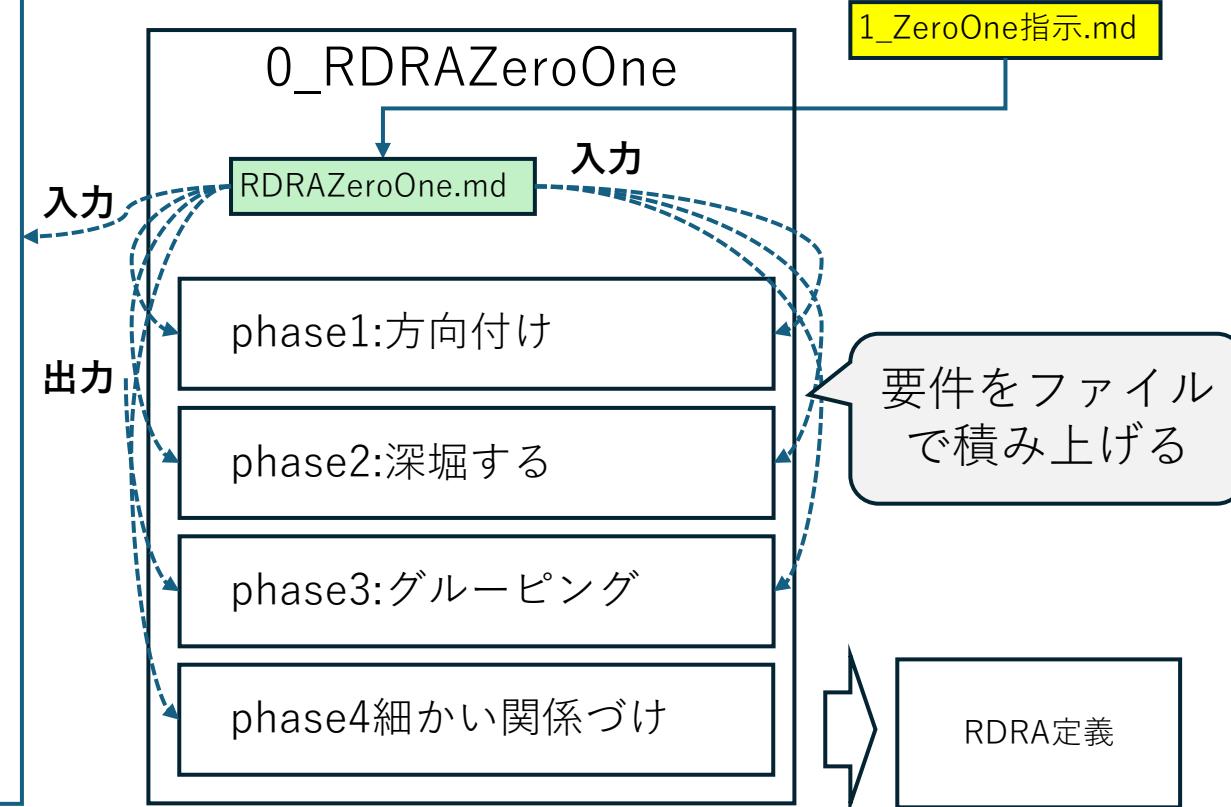
# 初期要望からRDRA定義を組立てる

初期要望.txt

```
# システム要求
## 背景
介護業界では、介護会員の要望に合わせ、様々な法規の中でサービススタッフの多様な働き方でスケジュールを調整する必要がある
## 要求
・サービススタッフの柔軟な働き方に対応しつつ介護会員の要望に合わせたスケジュール調整を行える仕組みを持つ
## ビジネスピリシー
・各サービススタッフの要望に合わせたスケジュール管理を行う
・訪問介護事業で小規模事業社を複数持つので、そこを一元管理を行う
・サービススタッフの働き方は自由に設定できる
・介護会員の要望に合わせたスケジュール調整を行う
## ビジネスパラメータ
・介護施設が増えた場合にも簡単に対応できる
・サービススタッフの増減に素早く対応できる
## 業務概要
・介護会員の管理（名前、連絡先などの様々な属性を管理）
・訪問介護の実施記録を管理する
・新しいサービススタッフの教育
・介護費用の計算と請求及び回収
```

方向付け 深堀する グルーピング 細かい関係づけ

- LLMでRDRAを定義する
  - 初期要望.txtの内容からRDRA定義を生成する



# RDRAを定義する

Phase1に出力される

RDRAZeroOne

•

• Cursor

- ⇒ 「?」
- ⇒ 「1」

「1」の入力でPhase毎に実行する

```
File Edit Selection View Go Run Terminal Help
RDRAアドバイス.md - DevinCursor2 - Cursor
Generate a name for the chat
RDRAアドバイス.md
?

メニューを表示します。
■ZeroOne
1. フェーズ単位の要件定義：RDRAZeroOneをフェーズ毎に実行する
2. 一括要件定義：RDRAZeroOneの全フェーズを一機に実行する
3. 生成した成果物の削除：RDRAZeroOneとRDRAファイルを削除する
■RDRA
4. 生成したファイルをRDRAフォルダーにコピーする
5. RDRAGraphを表示：関連データを作成しRDRAGraphを表示
6. RDRA定義の妥当性検証：RDRA定義の妥当性を検証します
■RDRAgent : 仕様
7. 仕様の作成：論理データ構造の作成と画面の作成
8. プロトタイプ画面を表示する
9. 仕様の妥当性を検証する
実行したい番号を入力してください。

@ RDRAアドバイス.md
Plan, search, build anything
Agent Ctrl+I Auto
NOTEPADS
OUTLINE
TIMELINE
X @ 0 △ 0
```

RDRAアドバイス.md

# 疑似言語

前になぜ  
そのファ  
イルを作  
成するこ  
とが重要  
なのかを  
説明する  
Rule:Ret  
urnメッ  
セージに  
基づいて  
実行する  
Rule:Age  
ntの実行  
後に実行  
結果をど  
うするか  
を「出力  
後の対  
応」を元  
に説明す  
る

if 初期  
要望.  
txt  
exist {  
if  
初期  
要  
望.  
txt  
内  
容  
がブ  
ラン  
ク {  
return "初期  
要望が「初期

42  
43  
44  
45  
46  
47

# CursorのAgentで何ができる1：フェーズ実行

## 実行結果

状態モデル説明  
介護会員状態 介護会員の登録から退会までの状態を管理  
サービススタッフ状態 サービススタッフの登録から退職までの状態を管理  
スケジュール状態 訪問介護スケジュールの作成から完了までの状態を管理  
介護記録状態 介護記録の作成から確定までの状態を管理  
教育状態 教育計画から評価までの状態を管理  
費用状態 費用計算から回収完了までの状態を管理

Phase1：  
指示に従って各ファイルを生成

Phase1の実行が完了しました。以下のファイルを作成しました：

**作成したファイル一覧**

1. アクター..tsv - システムに関わる人の役割（介護会員、サービススタッフ、管理者等）
2. 外部システム..tsv - 外部システム（介護保険システム、給与システム等）
3. ビジネスピリシー..tsv - ビジネスピリシー（柔軟な働き方対応、一元管理等）
4. ビジネスパラメータ..tsv - ビジネスパラメータ（介護施設分類、働き方分類等）
5. 業務..tsv - 業務とBUC（介護会員管理、訪問介護計画、教育管理等）
6. 情報..tsv - システムで扱う情報（介護会員情報、スケジュール情報等）
7. 状態モデル..tsv - 状態モデル（介護会員状態、スケジュール状態等）

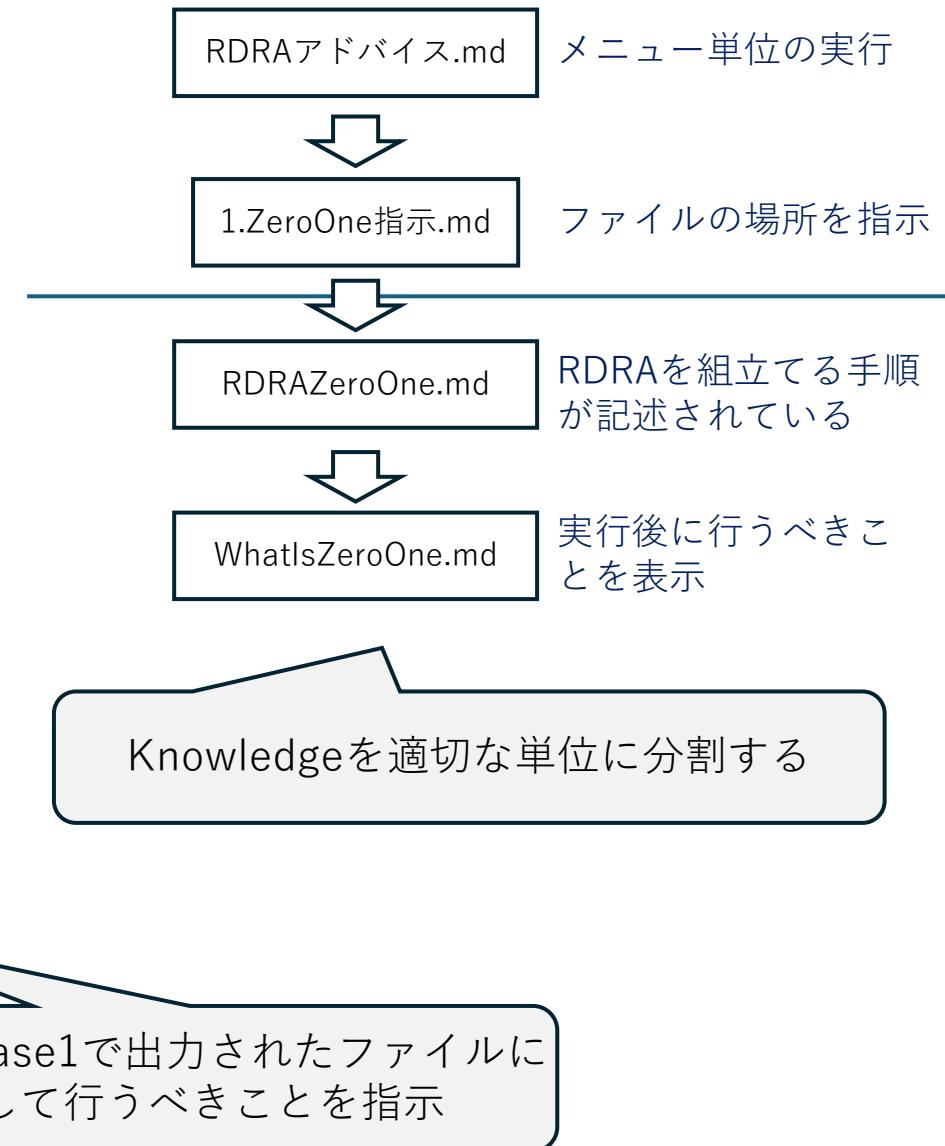
**出力後の対応**

これらのファイルは訪問介護システムの基本的な要件定義を表しています。次のステップとして、人が以下の作業を行う必要があります：

1. 内容の確認・修正：各ファイルの内容が初期要望に適切に反映されているか確認
2. 不足項目の追加：見落としがないか確認し、必要に応じて追加
3. 整合性の確認：各ファイル間の整合性を確認

確認・修正が完了したら、Phase2の実行に進むことができます。Phase2では、これらの基本情報を詳細化し、アクティビティ、ユースケース、条件、バリエーションを追加します。

## Knowledgeの分割



# CursorのAgentで何ができる：一括要件定義

RDRAアドバイス.md  
メニューを表示

こんにちは！現在のプロジェクトの状況を確認しました。  
このプロジェクトは介護業界向けのスケジュール管理システムの要件定義をRDRA（Requirements Definition using Reference Architecture）の手法で行うものです。

現在の状況を確認すると：

- Phase1のファイルは既に作成済み - 0\_RDRAZeroOne/RDRA/phase1/ に7つのファイルが存在
- Phase2以降は未実行 - phase2/, phase3/, phase4/ フォルダーは存在するが、ファイルは未作成

以下のメニューから実行したい作業をお選びください：

**ZeroOne**

- フェーズ単位の要件定義 - RDRAZeroOneをフェーズ毎に実行
- 一括要件定義** - RDRAZeroOneの全フェーズを一括実行
- RDRAフォルダーにコピー - Phase3,4のファイルをRDRAフォルダーにコピー
- RDRAGraphを表示 - 関連データを作成しRDRAGraphを表示
- 生成した成果物の削除 - RDRAZeroOneとRDRAファイルを削除

**RDRA**

- RDRA定義の妥当性検証 - RDRA定義の妥当性を検証

**RDRASpec (仕様)**

- 仕様の作成 - 論理データ構造の作成と画面の作成
- プロトタイプ画面を表示 - プロトタイプ画面を表示

どの番号の作業を実行しますか？

「？」 「2」  
一括要件定義を実行

- Cursor

- RDRA定義をAgentで行う
- 全Phaseを一括で実行する
  - RDRAアドバイス.md
    - ⇒ 「？」 メニューの表示
    - ⇒ 「2」 一括要件定義

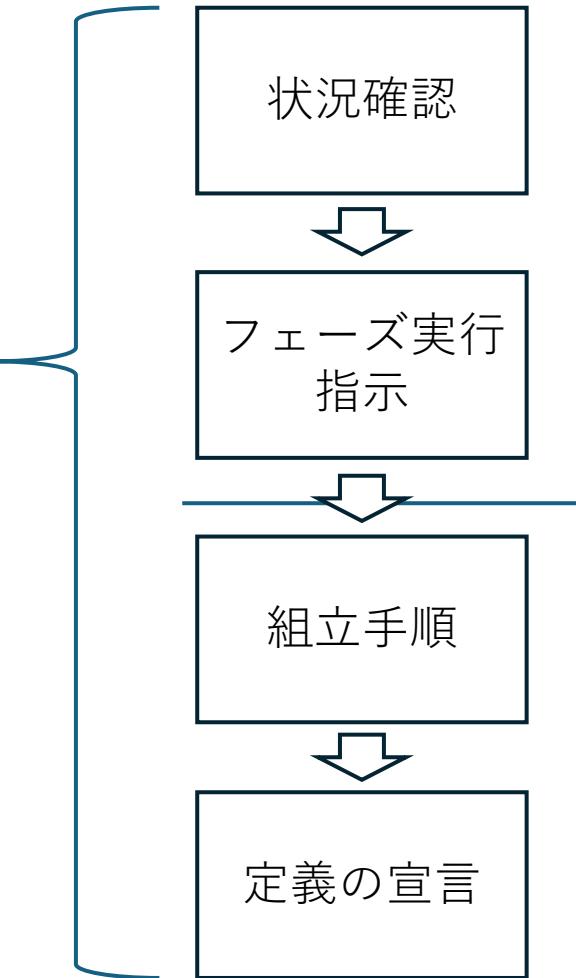
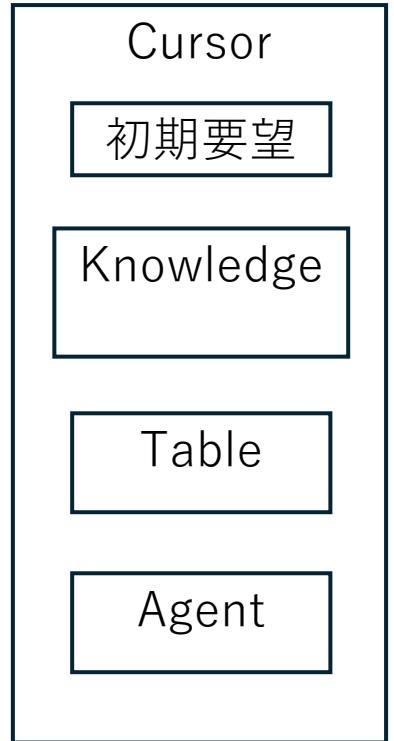
- Claude Codeは動く
- その他のモデルでは動かない

- Update Todos
  - 初期要望.txtの確認と読み込み
  - Phase1: ビジネスと情報分析の実行
  - Phase2: 詳細ビジネス構造の実行
  - Phase3: ユースケース統合の実行
  - Phase4: 関係モデリングの実行
  - RDRAファイルのコピーと検証

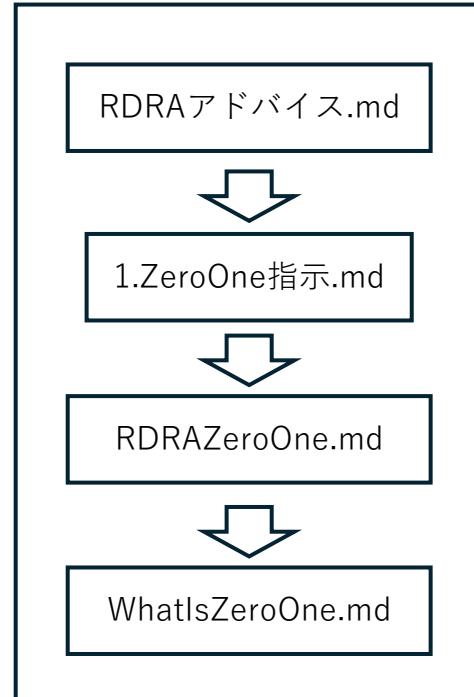
Demo.zipはClaude Codeで作成したもの

# ナレッジベースの構築

# Knowledgeの種類



- 状況確認
  - 生成した成果物の状況を把握し、次に行うことを探査
- フェーズ実行指示
  - 実行手順の指示
  - フォルダーとファイルの位置
- 組立手順
  - 要件の組立て方
  - 仕様の組立て方
- 定義の宣言
  - 要件
    - RDRAとは
    - RDRAの定義方法（表形式のRDRA）
  - 妥当性検証
  - 行るべきことの指示



# 定義：RDRAとは

RDRA.md

## # RDRA構造の説明

- ・「アクター」：システムに関わる人の役割を表す
- ・「アクターチーム」：同じような役割を持つ「アクター」をまとめ、組織の役割を示す場合もある
- ・「外部システム」：システムと連携する外部のシステムを表す

～

## # モデル間のつながりの規則

- ・「業務」は複数の「BUC」を配下にもつ
- ・「BUC」は複数の「アクティビティ」を配下にもつ
- ・「アクティビティ」(仕事)を行う「アクター」がつながる

～

## # 定義内容は各モデルのオブジェクトとして定義する

以下に例を示す

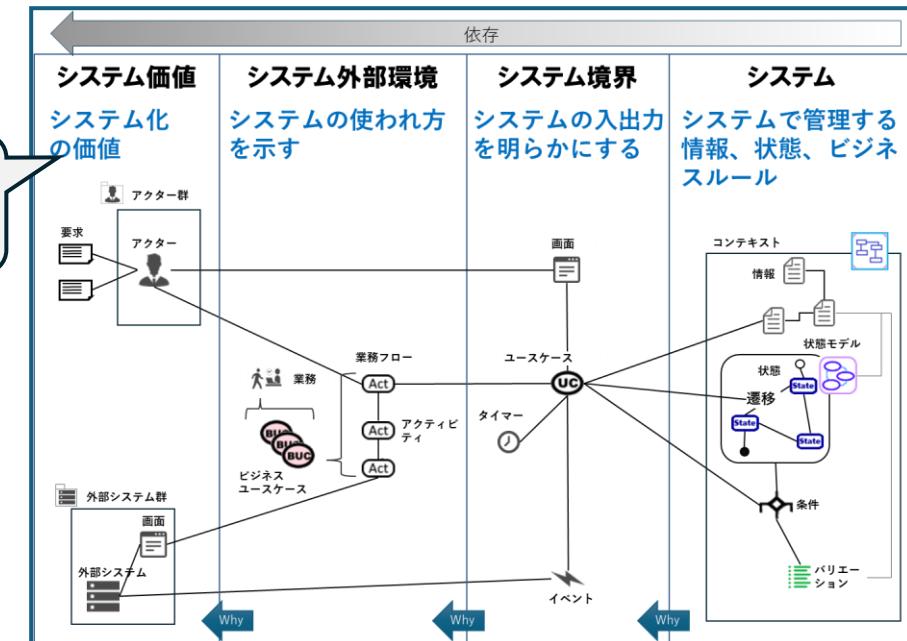
「会員」：モデルがアクター、オブジェクトが「会員」になる

「受注」：モデルが情報、オブジェクトが「受注」になる

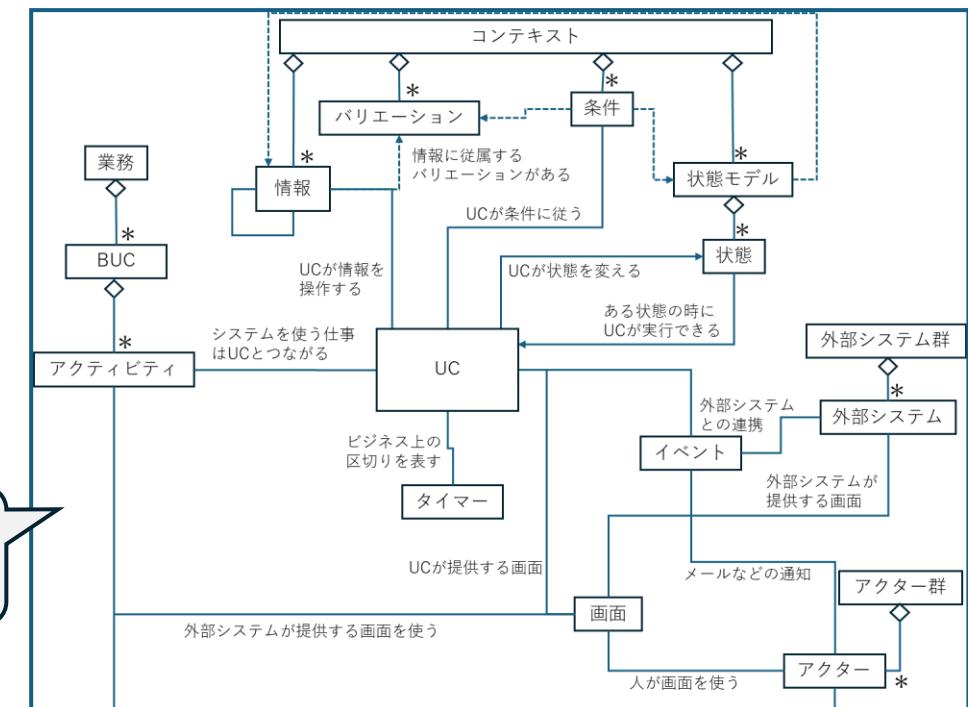
「受注を登録する」：モデルがUC、オブジェクトが「受注を登録する」になる

メタモデルを言語化

RDRAは定義する要素  
が決められている



RDRAはメタモデルで  
定義されている



# 定義：RDRA定義のフォーマット定義

RDRASheet.md

## # 表形式のRDRAモデリング

RDRASheetは表形式で要件定義を行うための形式です  
要件定義はシステムの可視化と同じです  
ファイル内では階層構造を表すために空白セルは上位のセルの値を引き継ぐことで階層構造を表現する

## ## 定義ファイルの特徴とメリット

### ### 1. 表形式のメリットと表現方法

- #### - \*\*表形式によるシステム可視化\*\*

表形式では、同じ行に並ぶ項目同士が関連しているとみなすため、ダイアグラムでの配置や線による関係性の明示よりも、要件の定義に集中できるというメリットがあります。これにより、最初は大まかなアイディアを素早く全ファイルに記入し、後から関連性を洗練させていくことが容易になります。

~

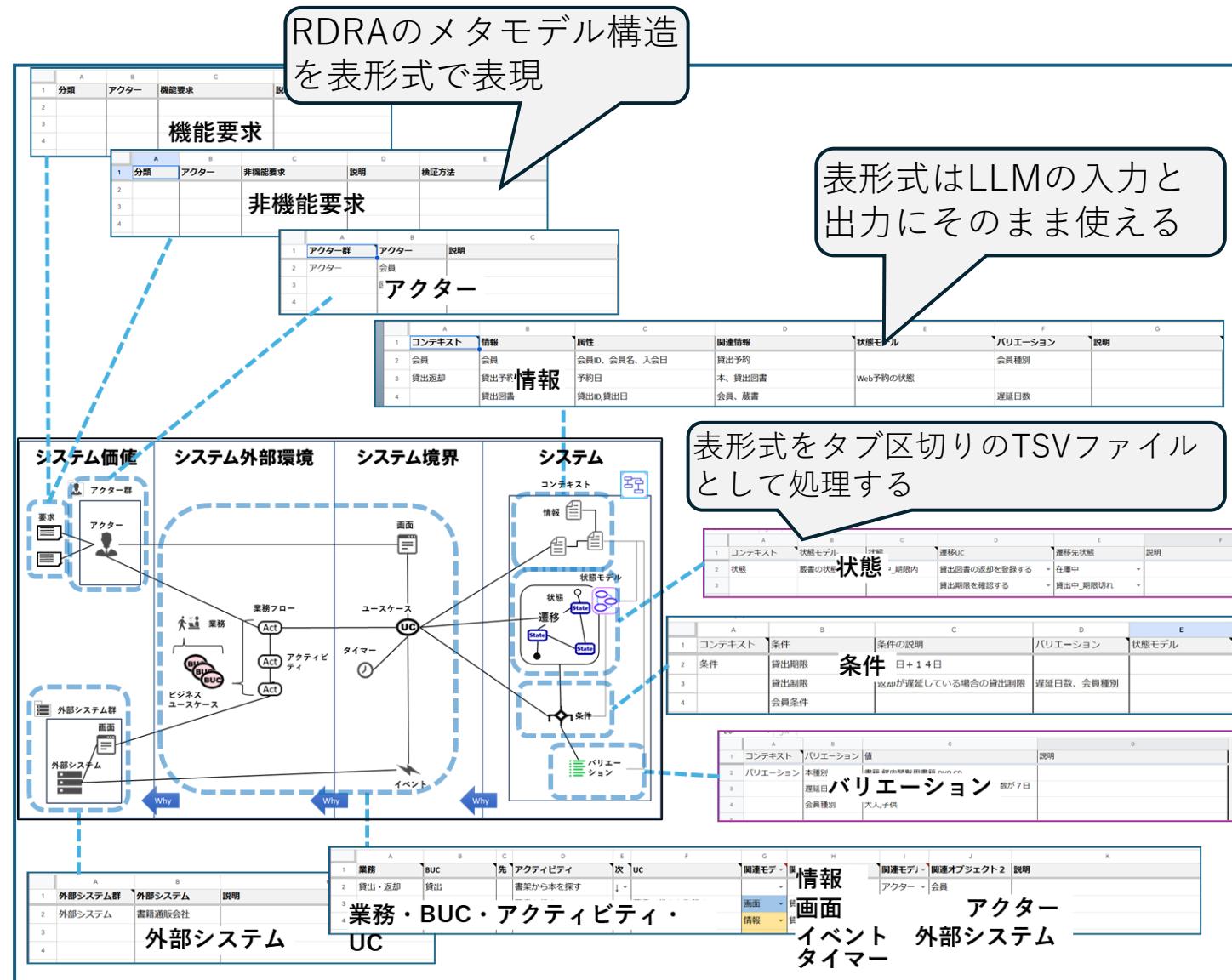
## ## 各ファイルの構造

表形式のRDRA定義は以下の5つのファイルで定義を行う。

- ・「アクター」ファイル
  - ・「外部システム」ファイル
  - ・「BUC」ファイル

818

## 表形式を言語化する



# 組立手順：4フェーズでRDRAを組立てる

RDRAを4フェーズで要件を組立てる

RDRAZeroOne.md

# RDRAで要件定義を行うためのフェーズ  
## \*\*RDRAZeroOneは段階的に要件を定義するために4つのフェーズです\*\*  
各フェーズでモデルを洗い出すときは「RDRA構造の説明」を十分に理解し洗い出す

## ## フェーズ1

### フェーズ1の目的

### 目的を達成するために次のことを行ってください

## ## フェーズ2

### フェーズ2の目的

### 目的を達成するためフェーズ1の結果を元に以下の～～

## ## フェーズ3

### フェーズ3の目的

### 目的を達成するためフェーズ2の結果を元に以下のStepを実行してください

#### Ste1: フェーズ2の「情報.tsv」「状態.tsv」「条件.tsv」「バリエーション.tsv」を元に以下の指示に従ってください  
～～

## ## フェーズ4：

### フェーズ3の目的

### 目的を達成するためフェーズ3の結果を元に以下の～～

方向付け

深堀する

グルーピング

細かい関係づけ

	フェーズ1	フェーズ2	フェーズ3	フェーズ4	RDRA
ビジネス	出発点として基本要素の洗出し  アクター役割 外部システム役割  要求業務 BUC 説明	基本要素を付けし、ビジネスルールを洗い出す  要求業務 BUC アクティビティ UC 説明	グルーピングとUCを中心に関係を組み立てる  アクター群 説明 外部システム群 外部システム説明  業務 BU アクティビティ UC 情報 条件 アクター 外部システム説明	要素間の関連付け  業務 BU アクティビティ UC 関連モデル1 関連オブジェクト1 関連モデル1 関連オブジェクト1 説明	BUC
ビジネスルール	ビジネスポリシー 説明  ビジネスパラメータ 説明	条件 説明  バリエーション 値 説明	コンテキスト 条件 説明  コンテキストバリエーション 値 説明	コンテキスト 条件のバリエーション 状態モデル  コンテキスト 条件のバリエーション 状態モデル	条件 バリエーション
システム	情報 説明  状態モデル 説明	情報 属性 関連情報 活用法  状態モデル 遷移前状態 遷移後状態 説明	コンテキスト 情報 属性 関連情報 活用法  コンテキスト状態モデル 遷移前状態 遷移後状態 説明	コンテキスト 情報 属性 関連情報 状態モデル  コンテキスト 状態モデル 状態 遷移UC 遷移元状態 説明	情報 状態

フェーズ毎にコンテキストを積み上げていく

4フェーズの組立を言語化する

# 定義 : WhatIsZeroOne

```
# RDRAZeroOneは以下の理由で4つのフェーズですすめている
## 4 フェーズで進める意味
## フェーズ 1 の位置づけ
### 初期要望.txtの位置づけ      ~      ### 業務の位置づけ

## フェーズ 2 の位置づけ
### 業務の位置づけ ~ ### バリエーションの位置づけ

## フェーズ 3 の位置づけ
### 業務の位置づけ ~ ### 外部システムの位置づけ

## フェーズ 4 の位置づけ
フェーズ4の目的はフェーズ3で要件の組立が終わったので、より詳細な要件間の関係づけを行います

### 情報の位置づけ
情報に関する他モデルである、状態モデルとバリエーションを関連づけ、情報と他のモデルの関係を明らかにし、相互のオブジェクトの必要性を明確にする

### 状態の位置づけ
状態モデル内の状態を明確にする
最初の状態と最後の状態を明確にし、状態の遷移にユースケースを追加し、相互のオブジェクトの必要性を明確にする

### 条件の位置づけ
ビジネスルールとなる条件について他のモデルであるバリエーションの位置づけ
相互のオブジェクトの必要性を明確にする
```

LLMの出力に対して何に着目し、どのように直すのか

各フェーズ毎の出力に対して行うべきことを示す

## 出力後の対応

これらのファイルは訪問介護システムの基本的な要件定義を表しています。次のステップとして、人が以下の作業を行う必要があります：

1. **内容の確認・修正**：各ファイルの内容が初期要望に適切に反映されているか確認
2. **不足項目の追加**：見落としがないか確認し、必要に応じて追加
3. **整合性の確認**：各ファイル間の整合性を確認

確認・修正が完了したら、Phase2の実行に進むことができます。Phase2では、これらの基本情報を詳細化し、アクティビティ、ユースケース、条件、バリエーションを追加します。

[Review changes](#)



# 番外編 LLMが賢くなっている

LLMが賢くなつて思いもしない挙動が増えている

問題：LLMが空のファイルを作り出した

私：なぜ空のファイルを作ったのか

AI：ひな型を作成する設定になっている

私：何処に「ひな型を作成する」設定があるのか

AI：明確なひな型を作成する指示はありません

RDRAアドバイス.md

実行を強要する

～連続的に実行する

ファイルが無ければ生成するようにしている  
LLMが最適化しようとしている？

```
phase1 = 「0_RDRAZeroOne/RDRA/phase1」  
if phase1 not exist "アクター..tsv" {  
    return "アクターが定義されていないので「1_ZeroOne指示.md」Phase1を実行します"  
}else if phase1 not exist "外部システム.tsv" {  
    return "外部システムが定義されていないので「1_ZeroOne指示.md」Phase1を実行します"  
}else if phase1 not exist "ビジネスポリシー..tsv" {  
~~
```

LLMの出力を如何に理解するか

# 表形式からGraph形式に変換する

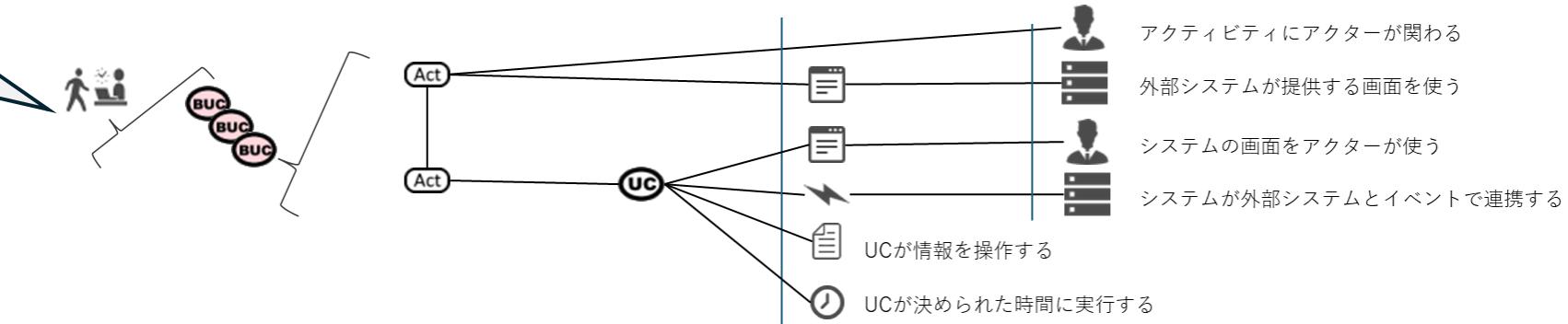
アクティビティ・UCにつながるオブジェクト定義											
A	B	C	D	E	F	G	H	I	J	K	
1 業務	BUC	先	アクティビティ	次	UC	関連モード	関連オブジェクト	関連モデル	関連オブジェクト2	説明	
2 貸出・返却	貸出		書架から本を探す	↓				アクター	会員		
3			蔵書を貸出す	▼	蔵書の貸出を登録する	画面	貸出登録	アクター	図書館員		
4				▼		情報	貸出図書				
5				▼		情報	蔵書				

表形式で定義するが構造を持っている

表形式からノードの関連へと変換する

複雑な処理はAgentではなくプログラムで変換

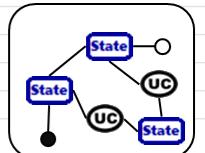
Agentはプログラムを実行する



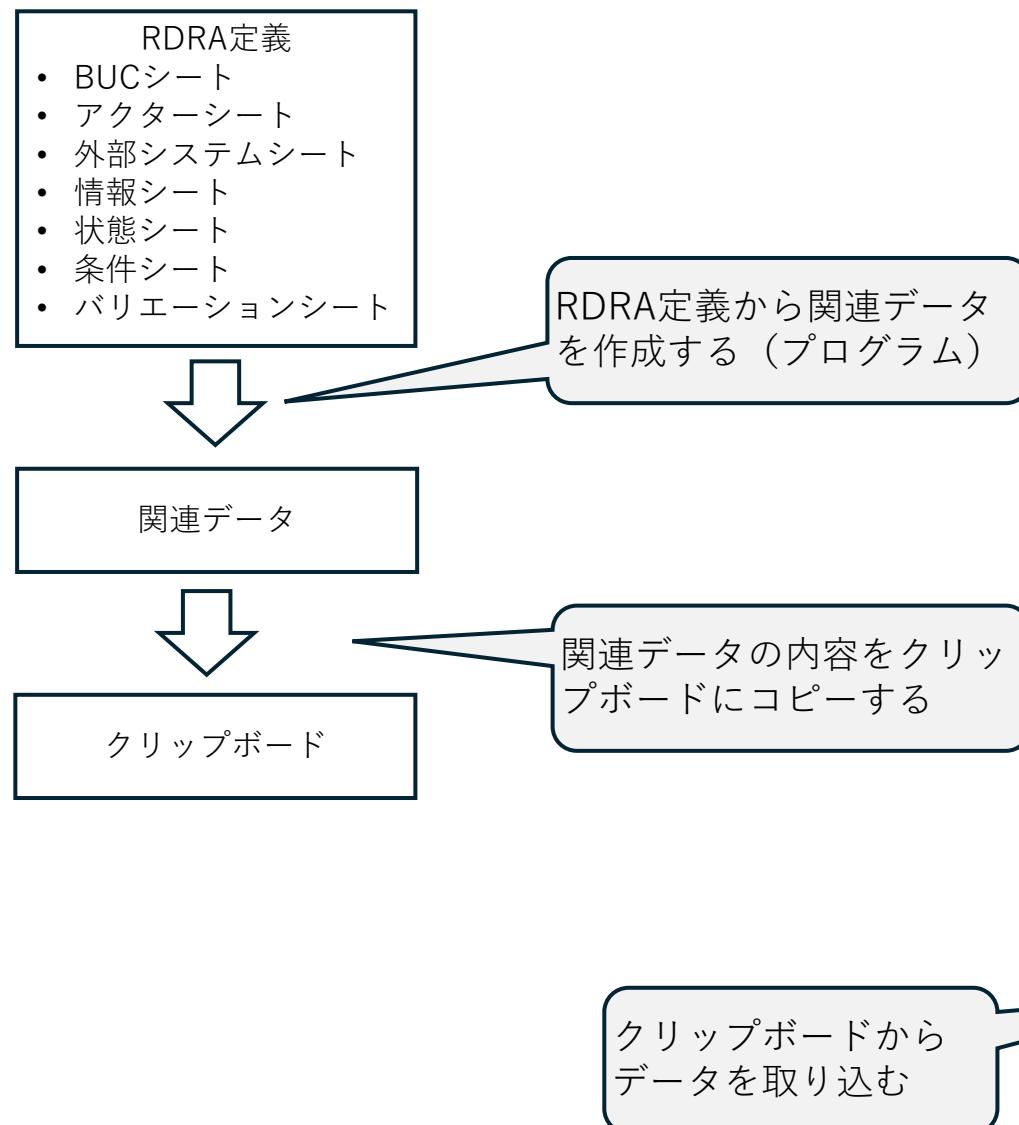
階層構造を表す				
A	B	C	D	E
1 コンテキスト	条件	条件の説明	バリエーション	状態モデル
2 条件	貸出期限	貸出日 + 14 日		
3	貸出制限	返却が遅延している場合の貸出制限	遅延日数、会員種別	
4	会員条件			

階層構造を表す						
A	B	C	D	E	F	
1 コンテキスト	状態モデル	状態	遷移UC	遷移先状態	状態モデル・状態の説明	
2 状態	蔵書の状態					
3						
4						
5						
6						

状態モデル



# ツール起動：定義情報をグラフィカルに表示する



- 行っていること

- 関連データの作成
  - RDRA定義からRDRAGraphのデータとなる、関連データを作成する
- 関連データの内容をクリップボードにコピーする
- RDRAGraphを表示



要件の妥当性を検証する

# 要件の妥当性検証

RDRA

妥当性検証環境

システムの導入環境に対して、有効な要件となっているかを検証する

入力

LLM

入力

- RDRA定義
- BUCシート
  - アクターシート
  - 外部システムシート
  - 情報シート
  - 状態シート
  - 条件シート
  - バリエーションシート

要件妥当性検証結果.md

出力

Validation.md

要件の検証方法を記述

妥当性検証環境.csv

妥当性を検証するためにシステム化対象の環境を入力する

	column 1	column 2	column	column 4	column 5
1	拠点	アクター	人数	制約	説明
2	事業所A 介護会員	200			
3		サービススタッフ	12		
4		請求担当者	1		
5	事業社B	介護会員	150	フルタイムのスタッフが少なく、スタッフの調整が難しいことがある	
6		サービススタッフ	10		
7		請求担当者	1		
8	事業社C	介護会員	190	サービススタッフが少なく、いつも忙しい	
9		サービススタッフ	11		
0		請求担当者	1		
1	ヘッドオフィス	介護会員	0	事業所の管理だけを行い介護作業は行わない	事業所を管理する拠点
2		サービススタッフ	2	各拠点でスタッフが足りないとき応援に行くスタッフ	
3		教育担当者	5		
4		請求担当者	3		

拠点ごとのアクターの制約から要件の検証を行う

# 要件の妥当性を検証する

要件妥当性検証結果.md

Preview 要件妥当性検証結果.md - DevinCursor2 - Cursor

Preview 要件妥当性検証結果.md X

## RDRA定義妥当性検証結果

### 検証概要

検証対象システム: 訪問介護システム  
検証実施日: 2025-07-24  
検証対象ファイル: アクター.tsv、外部システム.tsv、情報.tsv、状態.tsv、BUC.tsv、条件.tsv、バリエーション.tsv

### 総合評価

評価ランク: B (良好だが改善点あり)

RDRA定義は全体的に訪問介護システムとして適切に構造化されており、各ファイル間の関連性も適切に定義されています。ただし、実際の運用環境（妥当性検証環境.csv）を考慮すると、いくつかの現実的な制約に対する配慮が不足している点が見受けられます。

### 各ファイル別検証結果

#### 1. ファイル形式検証

合格項目

- 文字エンコーディング: 全ファイルがUTF-8形式で正しく保存されている
- ファイル形式: 全ファイルがTSV（タブ区切り）形式で正しく構成されている
- ヘッダー構造: RDRASheet仕様に準拠したカラム構造が適用されている

## BUC別実現可能性評価

### ● 高リスク: 緊急スケジュール対応

問題点:

- 事業所Cでは11名のスタッフで190名の介護会員をカバー（1人あたり約17名）
- 緊急事態発生時の代替スタッフ確保が困難

現実的な懸念:

- スタッフ1名が急病等で欠勤した場合、17名の介護会員のサービスに影響
- ヘッドオフィスの応援スタッフ2名では根本的解決にならない

改善提案:

- 緊急時のサービス優先度設定機能の追加
- 他事業所からの応援派遣プロトコルの明確化

### ● 中リスク: スタッフ教育

問題点:

- 教育担当者5名に対してサービススタッフ35名の教育は効率性に課題
- 事業所別の教育計画の調整が複雑

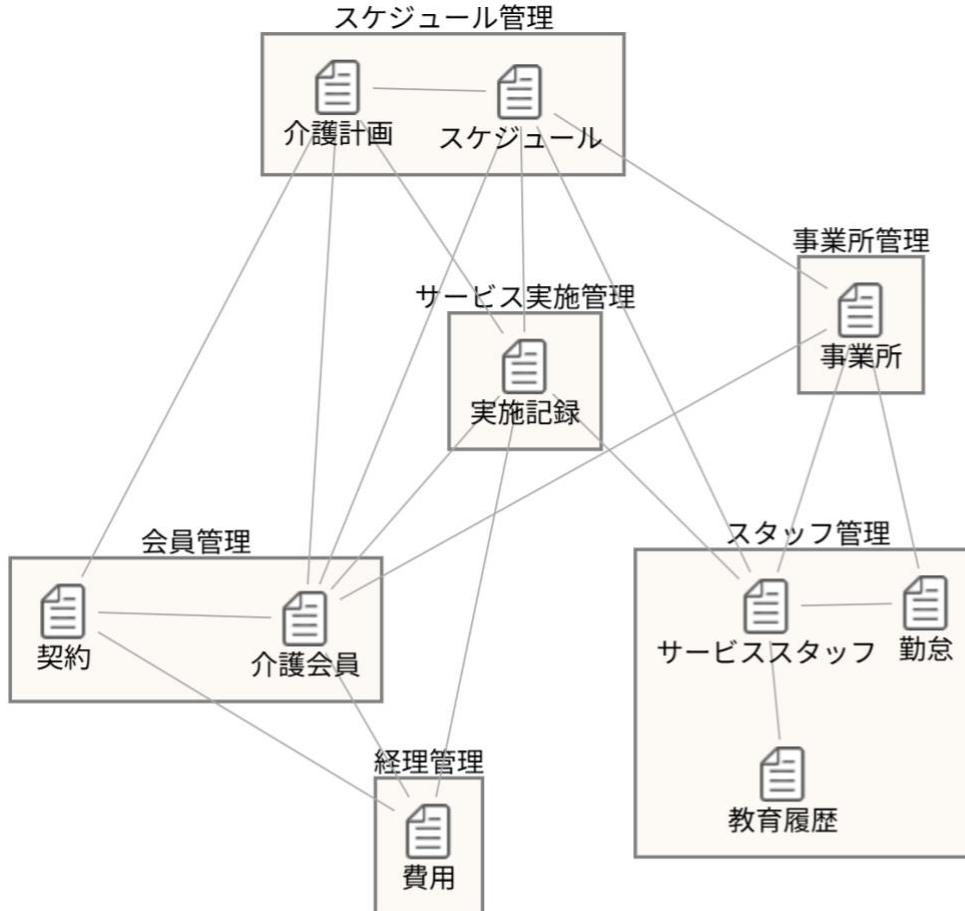
改善提案:

- オンライン教育システムの活用
- 事業所単位での教育責任者の設置

# 仕様の作成：2\_RDRASpec

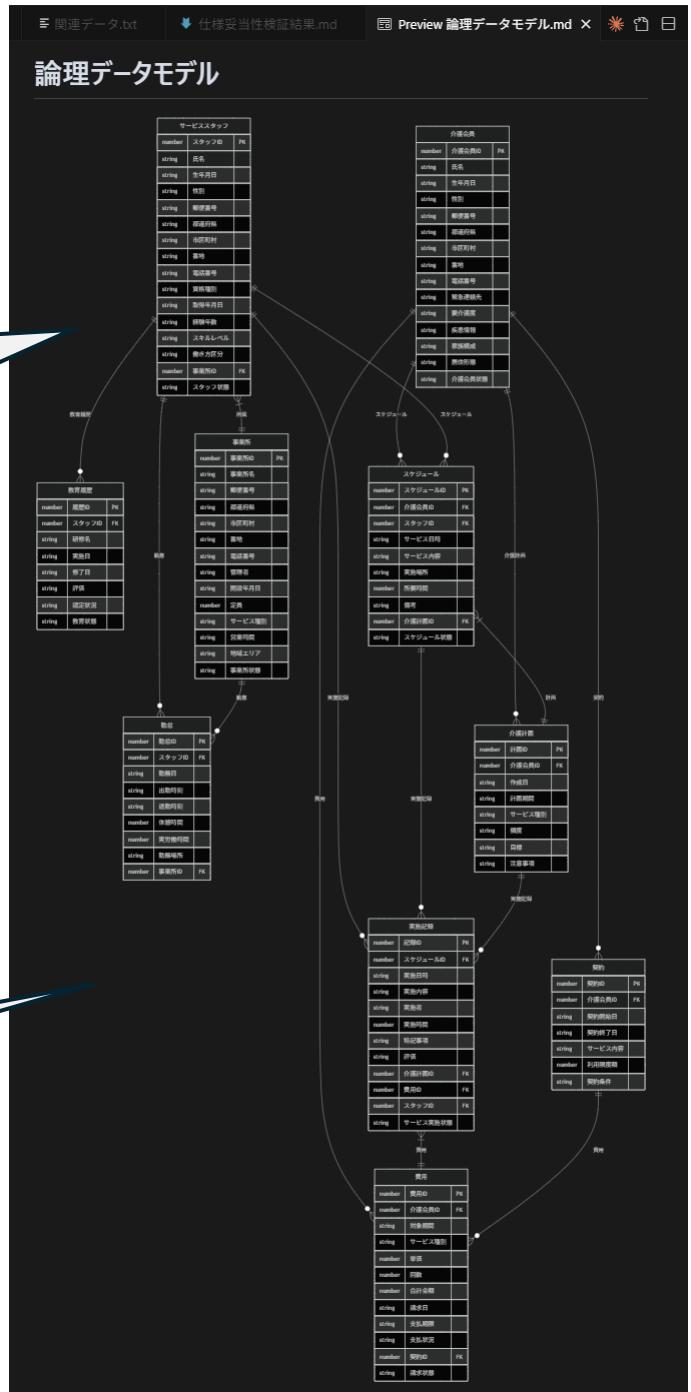
- 論理データモデル
- 画面の項目
- 条件 ⇄ 今後実現

# 論理データモデルの作成



RDRAの情報モデルから論理データモデルを作成する

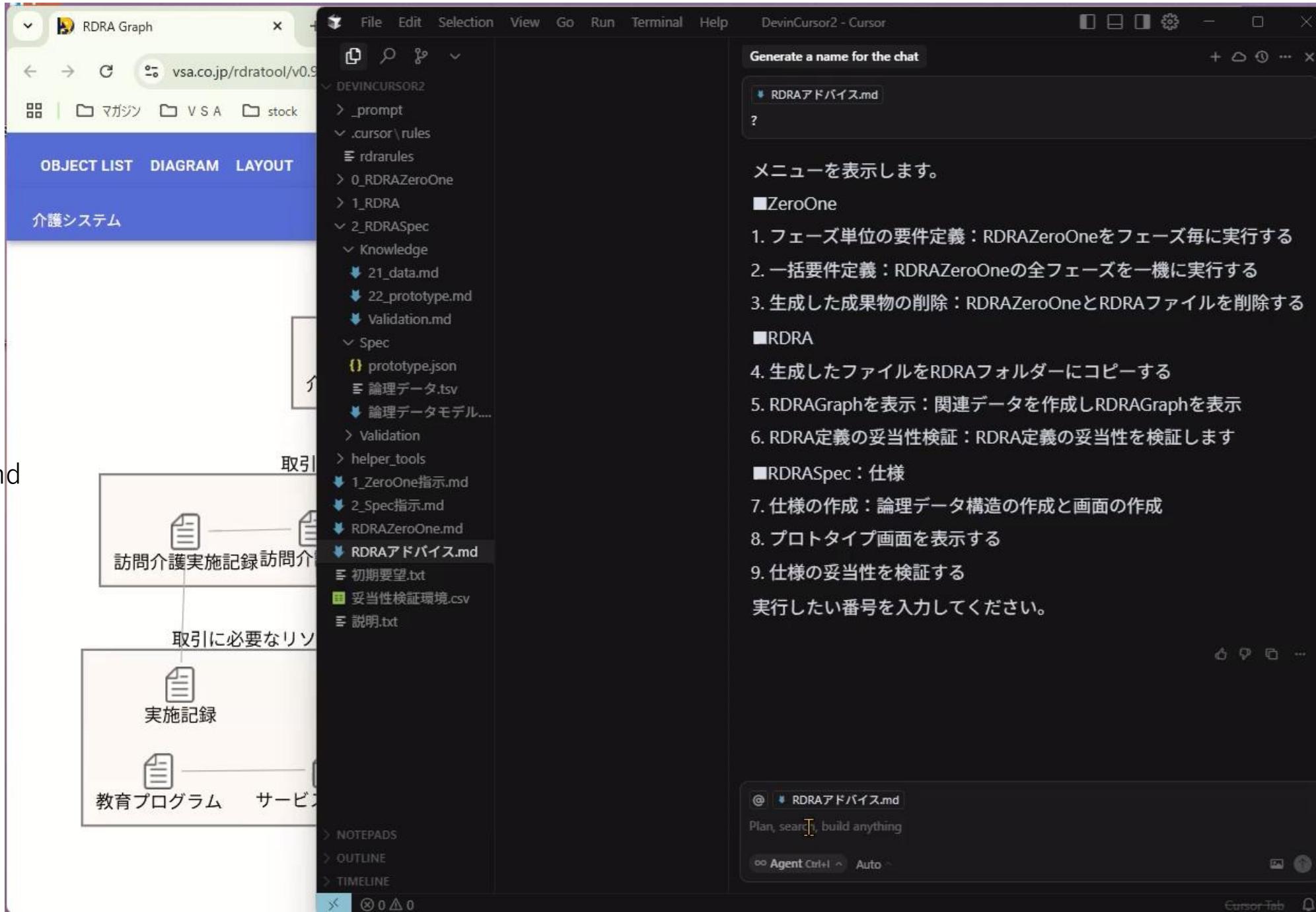
編集はMermaidで行う



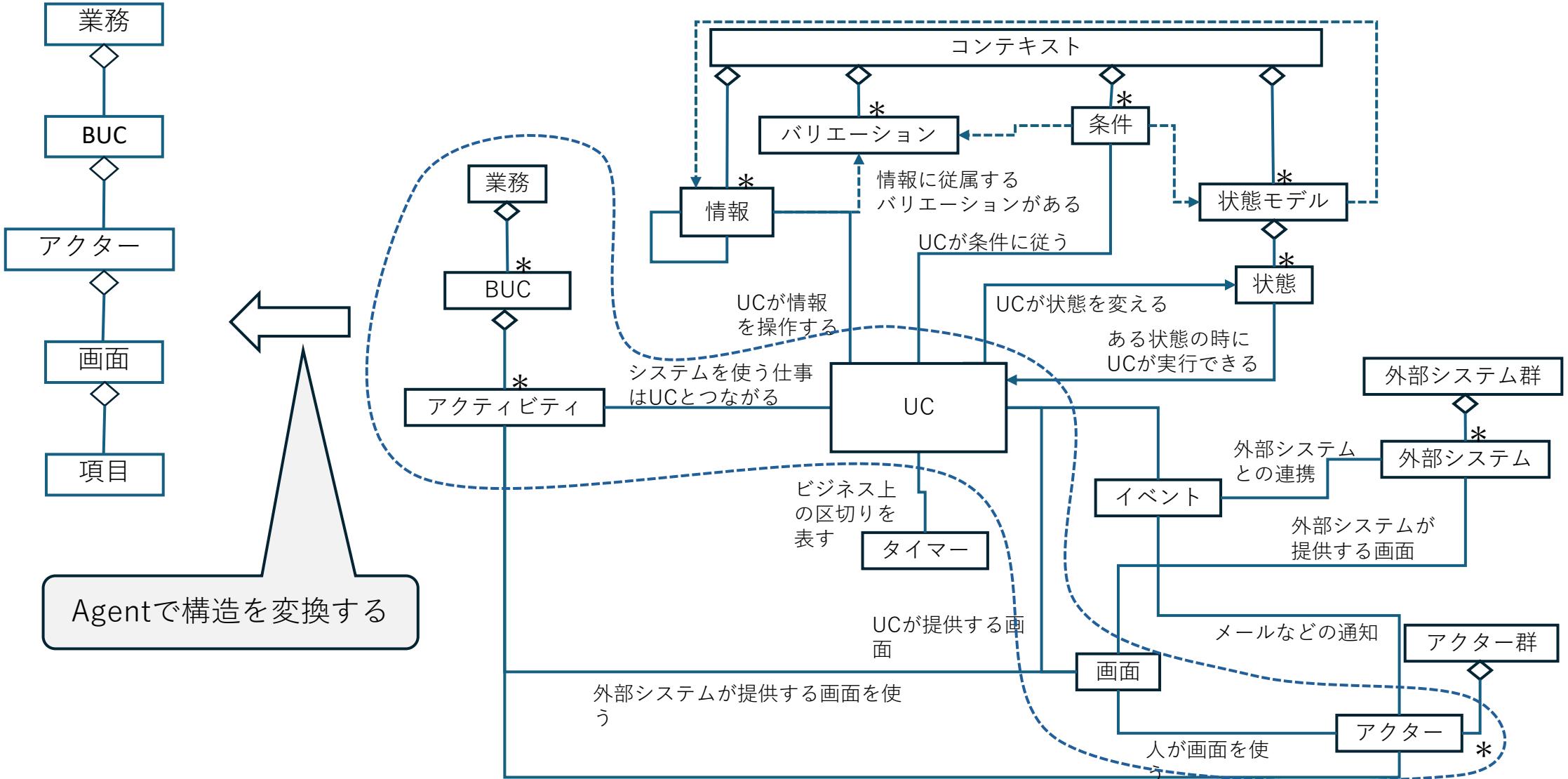
# 画面の項目の追加

## • Cursor

- @RDRAアドバンス.md
  - ⇒ 「？」
  - ⇒ 「8」



# 業務・BUCからアクター別の画面に構造変換する



# プロトタイプの画面項目エディタ

ターミナルで以下のコマンドでも実行できる

node .¥helper\_tools¥web\_tool¥prototype\_server.js

業務/BUC

介護会員管理

会員登録

会員情報更新

会員退会

介護記録管理

スタッフ教育管理

スケジュール管理

費用管理

プロトタイプ画面エディタ

管理者

会員登録画面

+ 項目追加 削除

会員ID 文字列 会員の一意識別子

名前 文字列 会員の氏名

住所 文字列 会員の住所

電話番号 文字列 会員の連絡先電話番号

要介護度 バリエーション 介護保険の要介護度

家族情報 文字列 家族の連絡先情報

要介護度確認画面

+ 項目追加 削除

会員ID 文字列 会員の一意識別子

要介護度 バリエーション 介護保険の要介護度

確認日 文字列 要介護度確認日

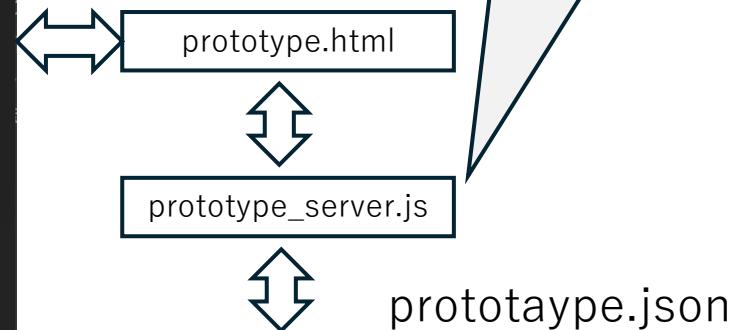
家族情報登録画面

+ 項目追加 削除

会員ID 文字列 会員の一意識別子

家族情報 文字列 家族の連絡先情報

Http-serverとして起動し  
prototype.htmlを起動  
prototype.jsonにアクセス

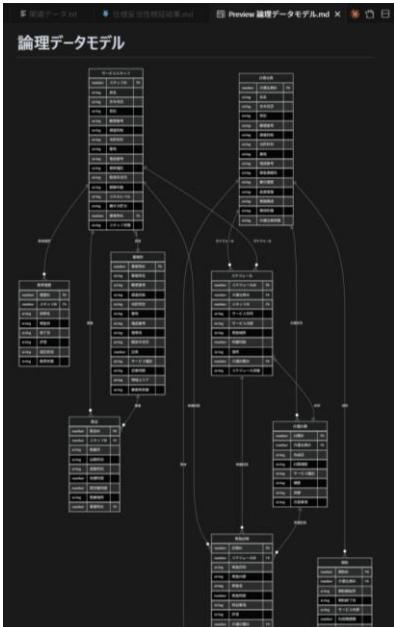


```
{  
  "businesses": [  
    {  
      "business": "介護会員管理",  
      "BUCs": [  
        {  
          "BUC": "会員登録",  
          "actors": [  
            {  
              "actor_name": "管理者",  
              "screens": [  
                {  
                  "screen_name": "会員登録画面",  
                  "fields": [  
                    ~~  
                  ]  
                }  
              ]  
            }  
          ]  
        }  
      ]  
    }  
  ]  
}
```

# 仕様の妥当性検証

# 仕様の妥当性検証結果

画面に関するある論理データで  
項目満たせるかを検証する



画面項目から論  
理データの項目  
を充実させる

論理データの項目の不足 w を指摘

## 仕様の妥当性検証結果

### 検証概要

本検証では、論理データモデル（論理データモデル.md）とプロトタイプ画面仕様（prototype.json）の遂行の観点から検証しました。

### 検証結果

#### 1. 画面と論理データの関係性の妥当性

✓ 整合性が取れている項目

- 介護会員管理: 画面項目と論理データモデルの介護会員エンティティの項目が一致
- 契約管理: 介護会員登録完了画面で契約エンティティの作成が適切に設計
- スケジュール管理: スケジュールエンティティと画面項目の対応が適切
- 実施記録管理: サービス実施画面と実施記録エンティティの連携が設計済み

⚠ 改善が必要な項目

##### 1. 住所情報の分割不整合

問題:

- プロトタイプ画面では「住所」として一括表示
- 論理データでは「郵便番号」「都道府県」「市区町村」「番地」に分割

影響: データ入力・表示の際の処理が複雑化

推奨改善:

```
// 収集画面では一括入力を許可し、入力画面で分割表示
"介護会員情報収集画面": {
  "住所": "string" // 一括入力
},
"介護会員情報入力画面": {
  "郵便番号": "string",
  "都道府県": "string",
  "市区町村": "string",
  "番地": "string"
}
```

#### 3. 論理データモデルの項目不足

⚠ 業務実行に必要だが不足している項目

##### 1. 介護会員エンティティ

-- 追加推奨項目  
利用開始日 DATE  
担当ケアマネージャー VARCHAR(100)  
医療情報 TEXT  
アレルギー情報 TEXT

##### 2. サービススタッフエンティティ

-- 追加推奨項目  
雇用形態 VARCHAR(50)  
時給 DECIMAL(10,2)  
交通費 DECIMAL(10,2)  
緊急連絡先 VARCHAR(100)

##### 3. スケジュールエンティティ

-- 追加推奨項目  
移動時間 INTEGER  
確認者ID INTEGER  
緊急度 VARCHAR(20)

# 環境依存部分はうまくいかない

- Windowsに依存する処理
  - ファイルの読み込み時のエンコード
  - コマンドが環境依存シェラーを起こす

この部分でエラーになる  
PowerShellで使えない設定になる

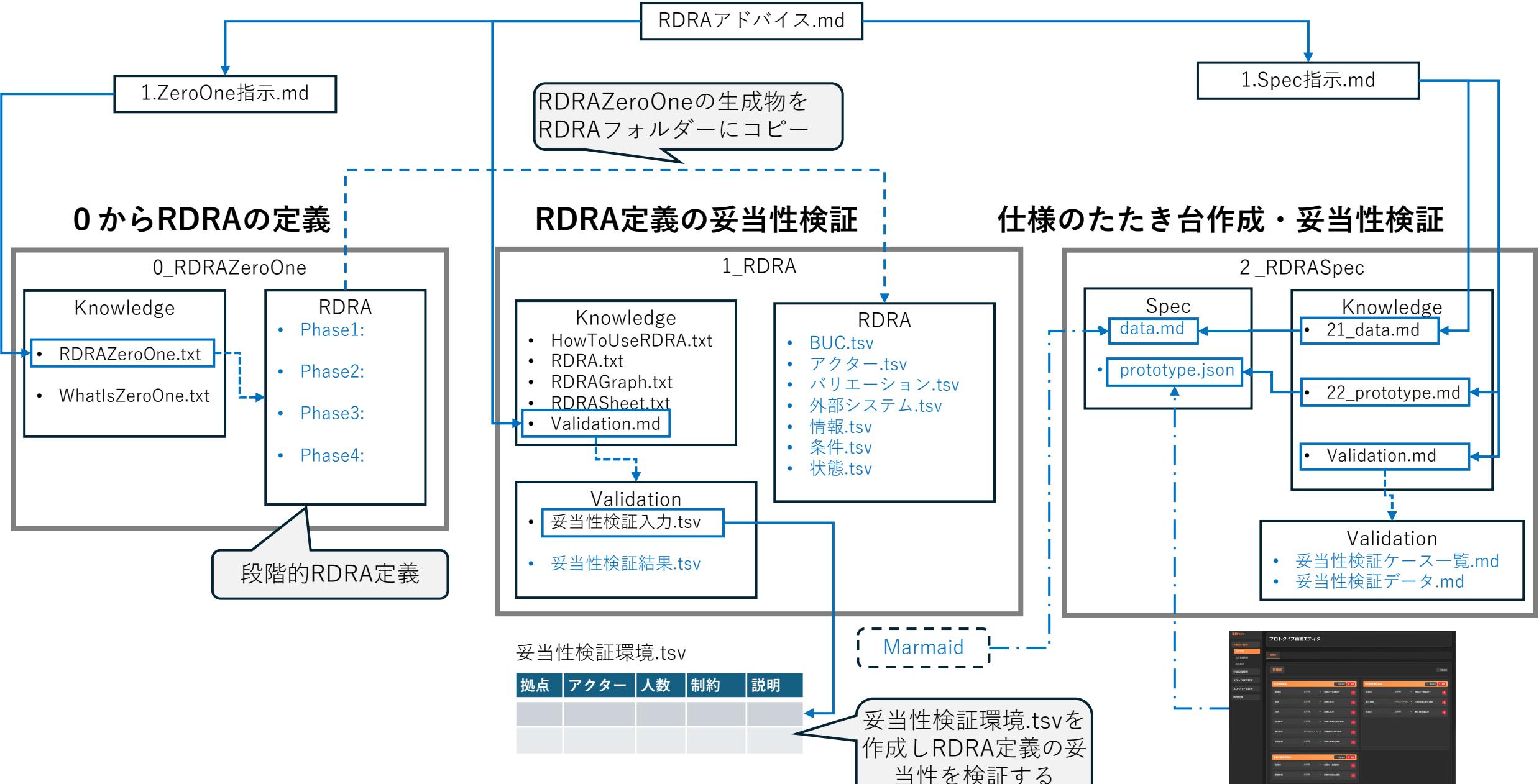
The screenshot shows a code editor interface with several tabs at the top: 'node helper\_tools/web\_tool/prototype\_server.js', 'runDeleteFiles.txt', 'showRDRAGraph.txt', and another tab partially visible. The main area displays a script named 'showRDRAGraph.txt' under the 'helper\_tools/command' directory. The script contains the following code:

```
1
2
3 powershell -Command Get-Content -Path 1_RDRA\関連データ.txt -Encoding UTF8 | Set-Clipboard
4
5 Start-Process "https://vsa.co.jp/rdratool/v0.93/index.html?clipboard"
6
7
8
```

A yellow oval highlights the line 'powershell -Command Get-Content -Path 1\_RDRA\関連データ.txt -Encoding UTF8 | Set-Clipboard'. A callout bubble points to this oval with the text 'この部分でエラーになる PowerShellで使えない設定になる' (This part causes an error. It's a setting that can't be used in PowerShell).

At the bottom right of the code editor, there is another callout bubble containing the text '実行コマンドをLLMに読ませて実行させる' followed by two options: '×Claude Code' and '△Grok4'.

# まとめ：CursorRDRAの構成



# まとめ：テストの文脈での応用

