要件定義は
何をどう定義するのか？

in JavaFesta　2009／10／30

Relationship driven requirement analysis　⇒　RDRA（ラドラ）

Who am I

- ㈱バリューソース
  - 代表取締役　社長
  - 神崎　善司
  - zkanzaki@vsa.co.jp

- 普段は
  - 要件定義のコンサルテーション
  - 要件定義やUMLのセミナー開催

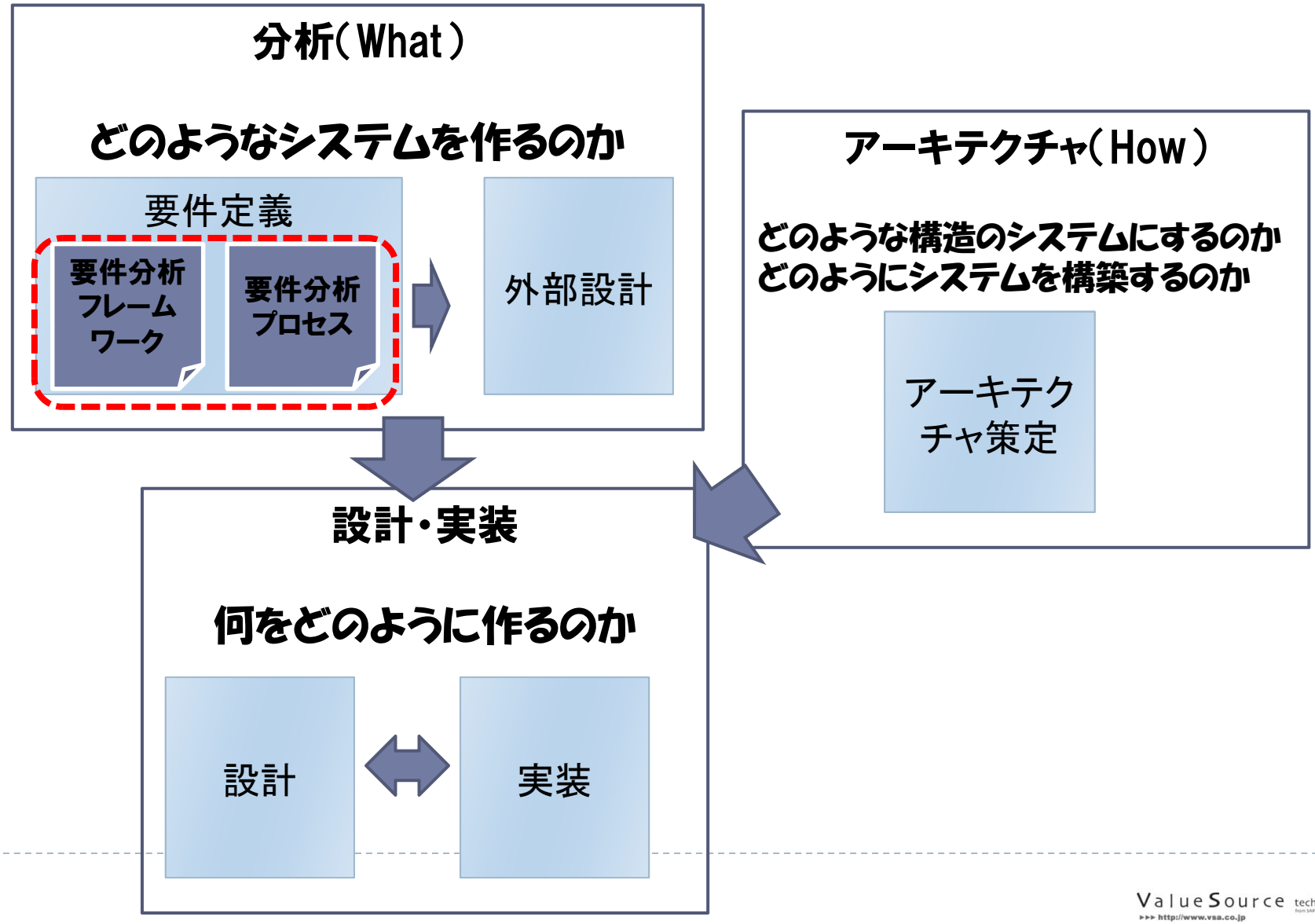
- 要件定義との関係は
  - 20年ほど前からオブジェクト指向を中心にシステム開発全般のコンサルティングを行う
  - 10年ほど前から上流工程を中心としたコンサルティングを行う
  - その間一貫してモデル中心で行う
  - その経験を活かしてモデルを使った要件定義の手法をまとめる

アジェンダ

- 内容
  - 上流工程は混乱しやすい
  - 混乱しないようにするためには
  - 要件には構造がある
  - 軌道修正しながら進める

- 持ち帰っていただきたいこと
  - 要件定義には構造があり、その構造を利用することでシステマティックに要件を定義できる
  - 軌道修正しながら全体を徐々に深掘りする

今日のプレゼンの位置づけ



上流工程では何が起こっているのか

そして　どうするの？

このようなことはないですか？

いつまでたってもシステムの全体像が見えてこない

ものすごいドキュメントが出来てくるが誰もシステムの目的、役割が分からない

分析という名の転記作業ばかりを行っている

会議が毎回ループして先に進まない

議論が直ぐに袋小路に入って先に進まない

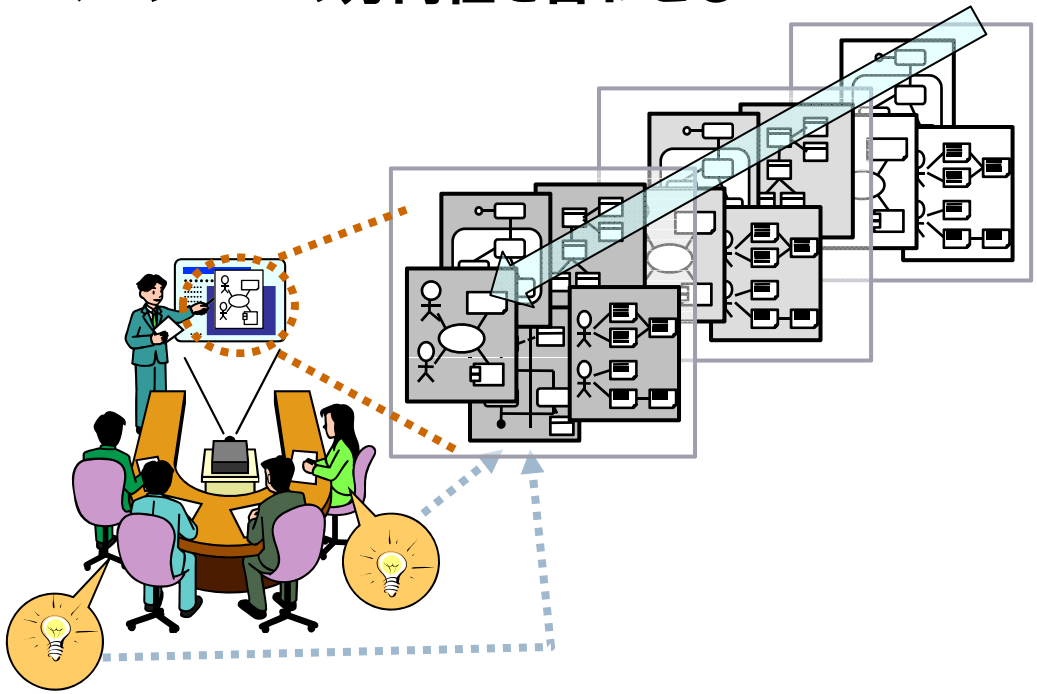
じゃあどうする？

議論を積み上げて要件としてまとめる

◆合意を積み上げていき議論に一貫性を持たせる

◆要件定義に筋道をつける

◆メンバーの方向性を合わせる



この言葉が続くようだと合意が得られていない証拠

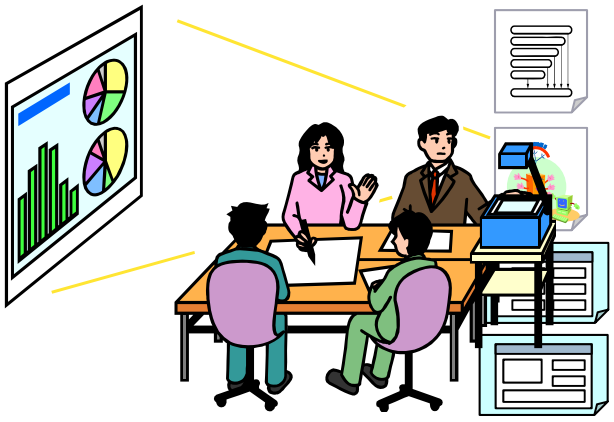
じゃあどうする？

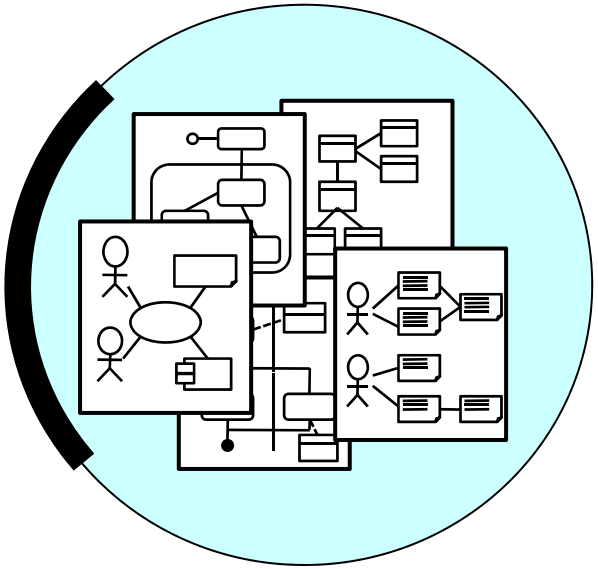
整合のとれた情報をもとに顧客をリードする

◆整合性のとれた資料をもとに、顧客の理解できる資料を作成する

◆顧客が理解出来る資料はごく一部

◆整合のとれた情報をもとに議論のぶれを少なくする



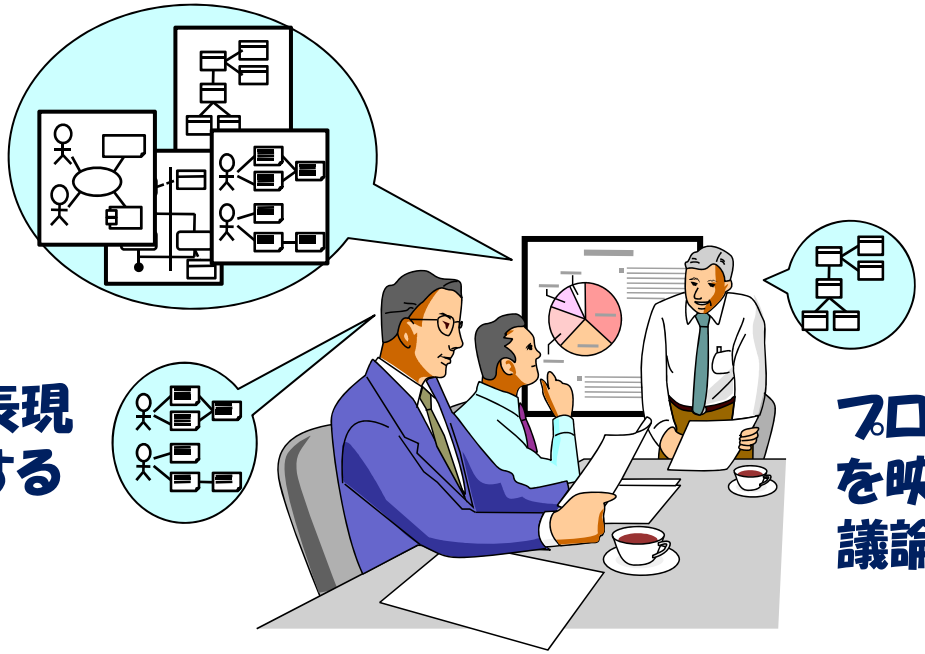


じゃあどうする？

アイディアを共有しコンセプトを得る

◆アイディアを視覚化し共有する

◆考える　合意する　理解する　そして説明可能になる



図的な表現を使用する

プロジェクターに資料を映しながらみんなで議論する

考え　→　合意し　→　説明できる

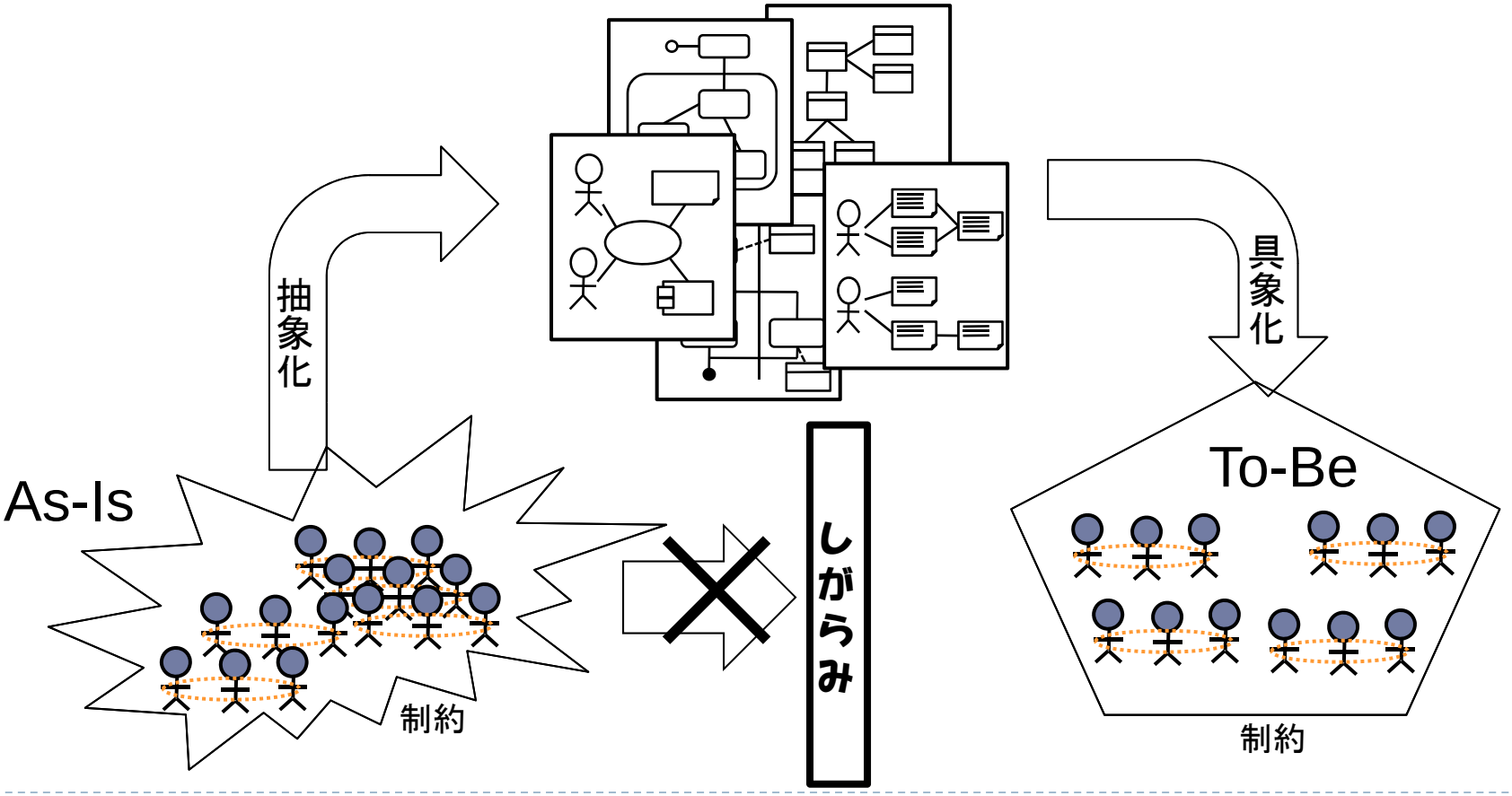
じゃあどうする？

一旦抽象化しToBeに落とし込む

議論が袋小路に入ってしまったら

◆一旦抽象化し本来あるべき姿を模索する

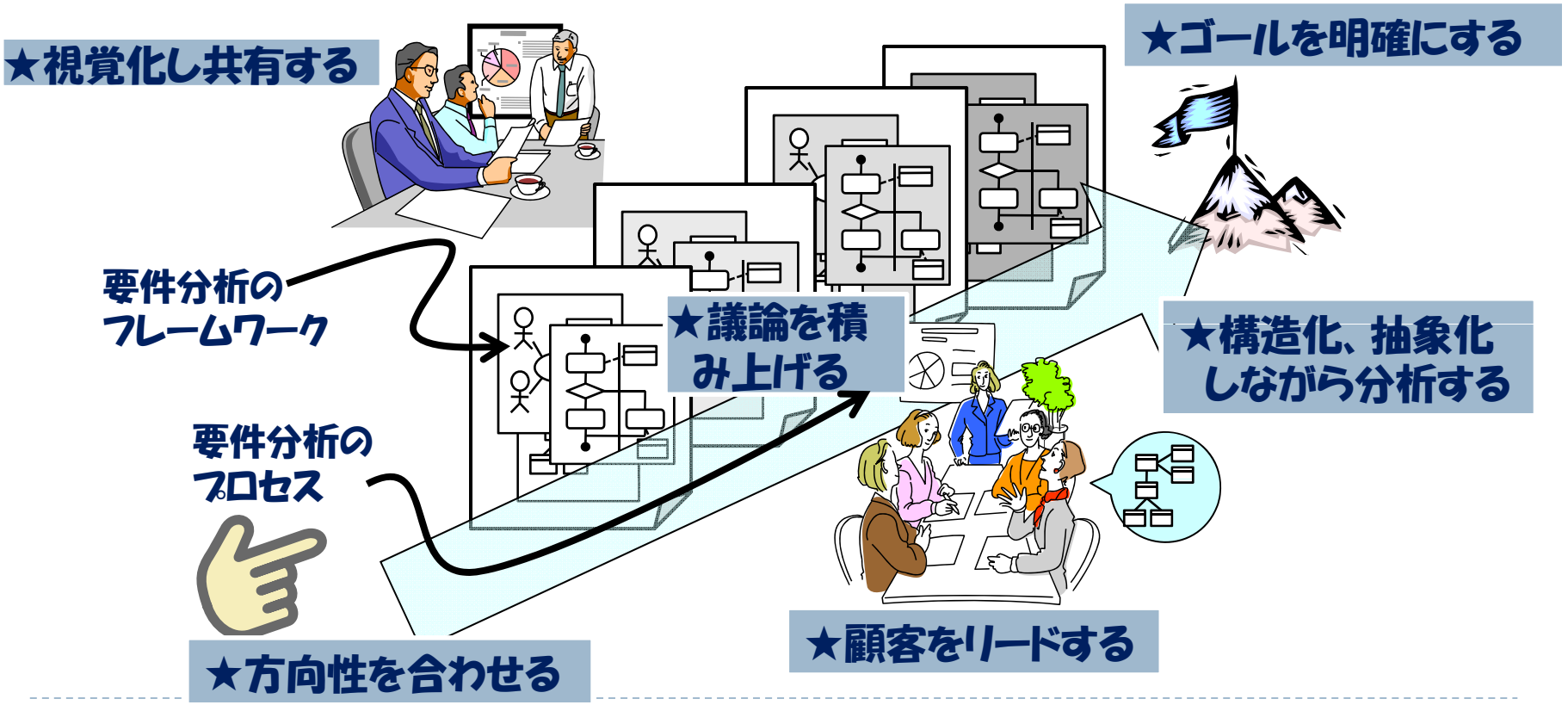
◆その上で今の制約にあわせて具体化する



結局　どうするの？

より混乱を少なく要件定義を行うためには……

ゴールを示し、そのための道筋を明らかにする。そして対象を視覚化する。
そして対象を構造化、抽象化して議論を積み上げながら要件を形にする。

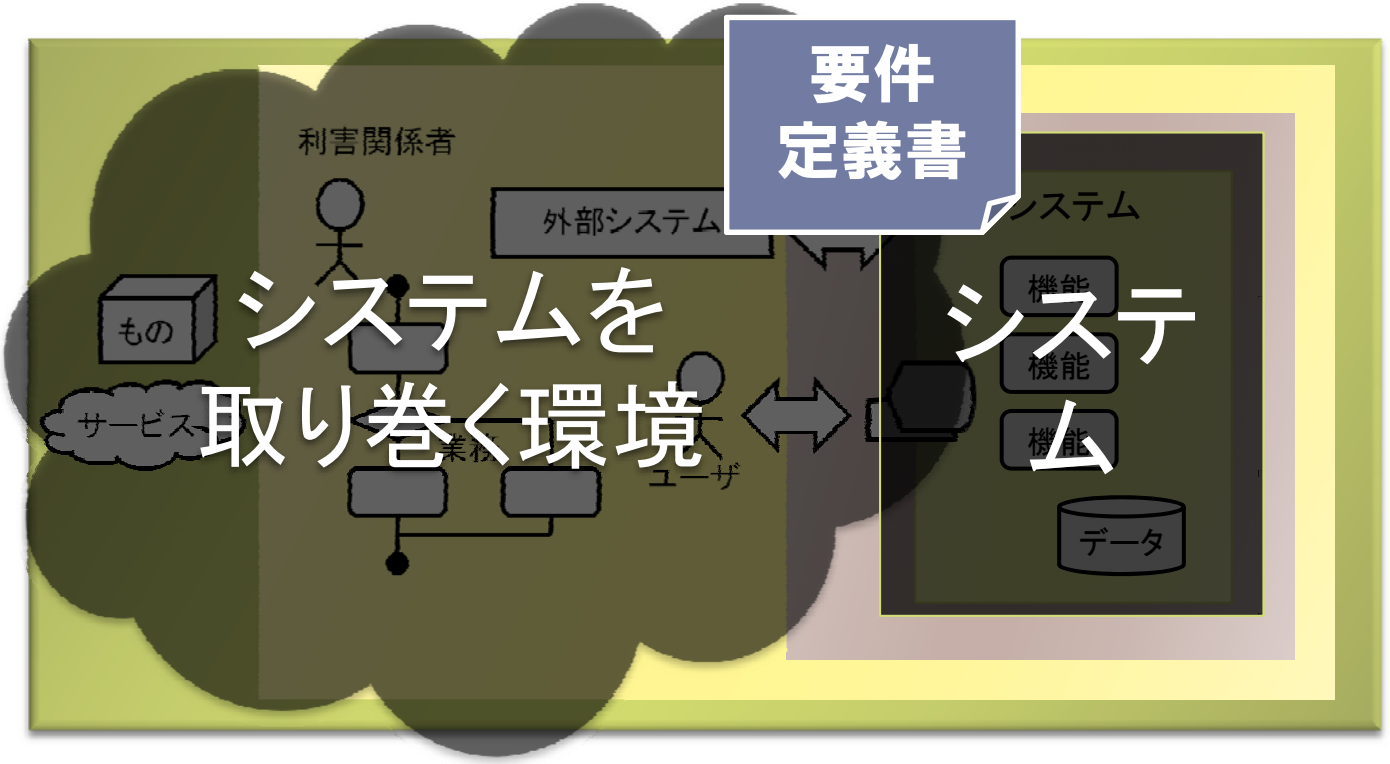


要件分析フレームワーク

要件定義の枠組み

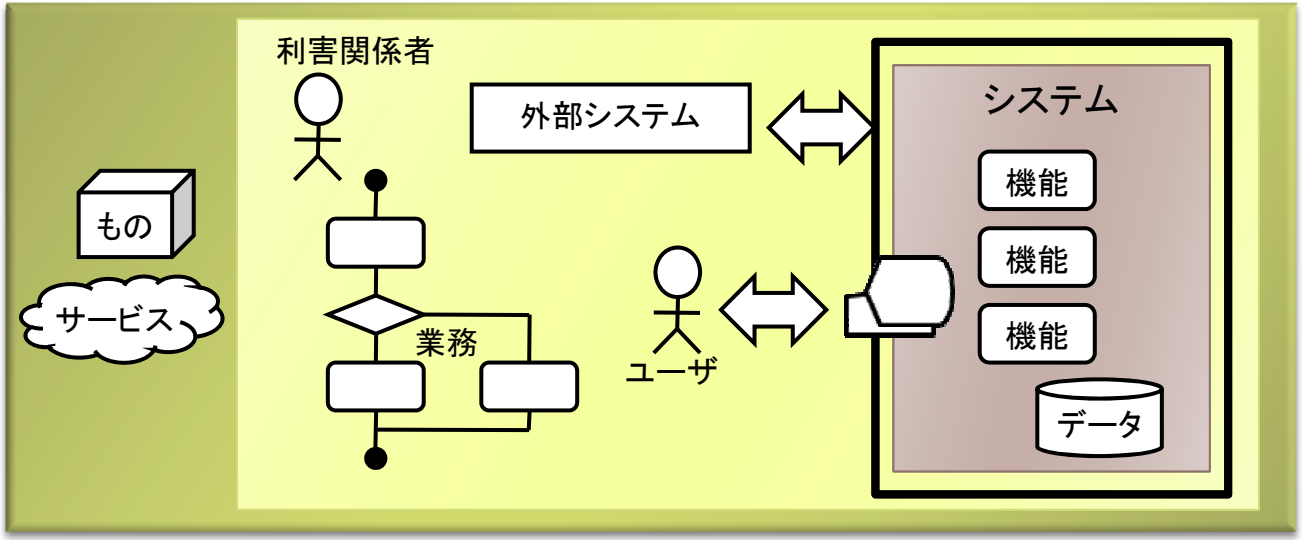
上流工程での混乱を減らすために要件定義の枠組みが必要

要件定義には何を定義すればいいのか



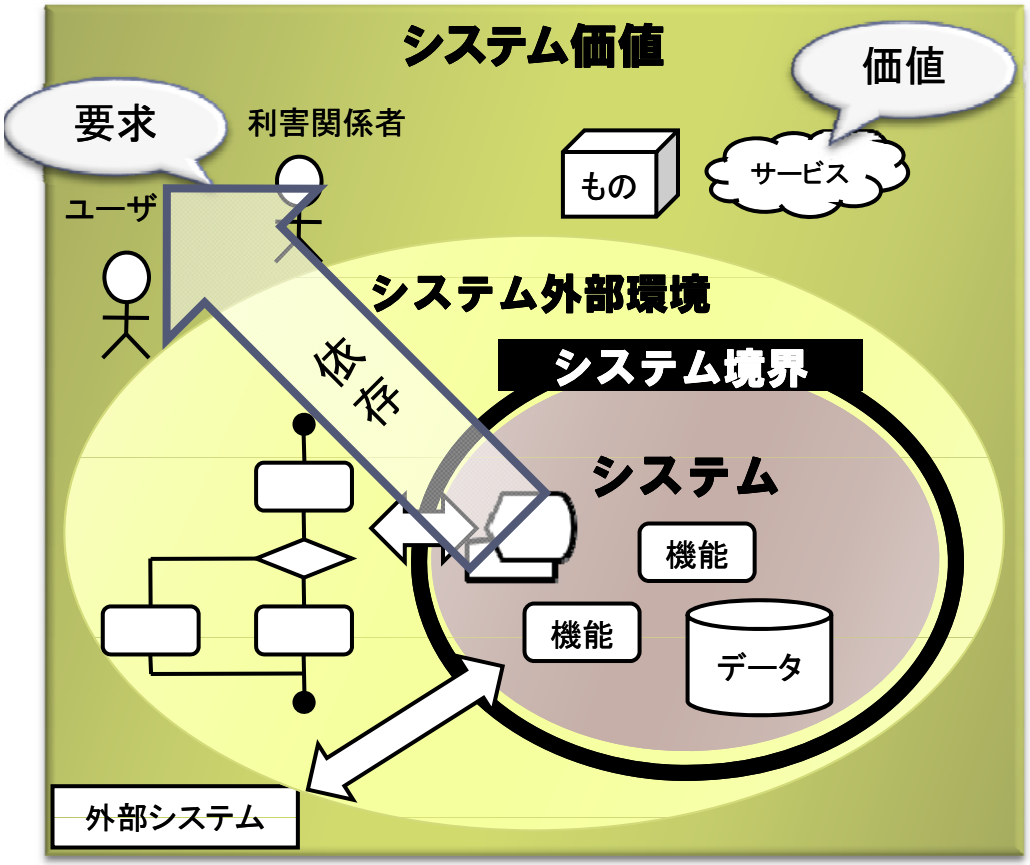
RDRAでは「要件定義の対象をシステムとシステムを取り巻く環境」と考える

要件定義では何が定義されないといけないのか



- このシステムの目的（価値）は？
- どのような人、外部システムと関わるのか？
- どのようにシステムは使われるのか？
- システムとの接点は？
- その時の入出力情報は？
- システムに必要な機能は？
- その機能が使用するデータは？

要件に枠組みを与える



1.【システムの価値】を捉える

- 対象業務に関わる人を洗い出す
- 関係する外部システムを洗い出す
- 要求を捉える

2.【システム外部環境】を捉える

- 対象業務の流れを捉える
- 対象業務に関わる概念を明らかにする
- システムの利用シーンを明らかにする

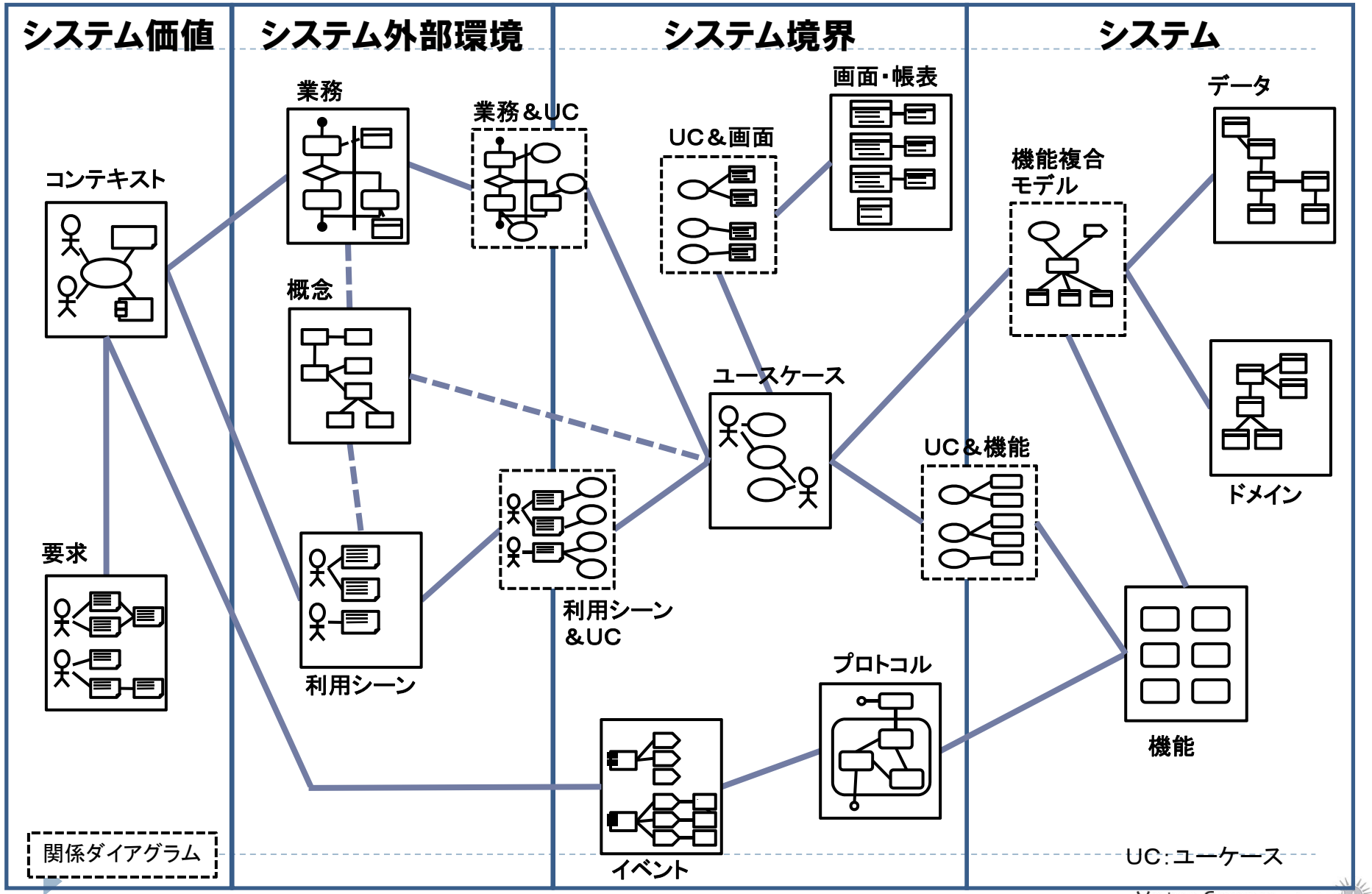
3.【システム境界】を捉える

- ユーザインターフェースを明らかにする
- 外部システムとのインターフェースを明らかにする

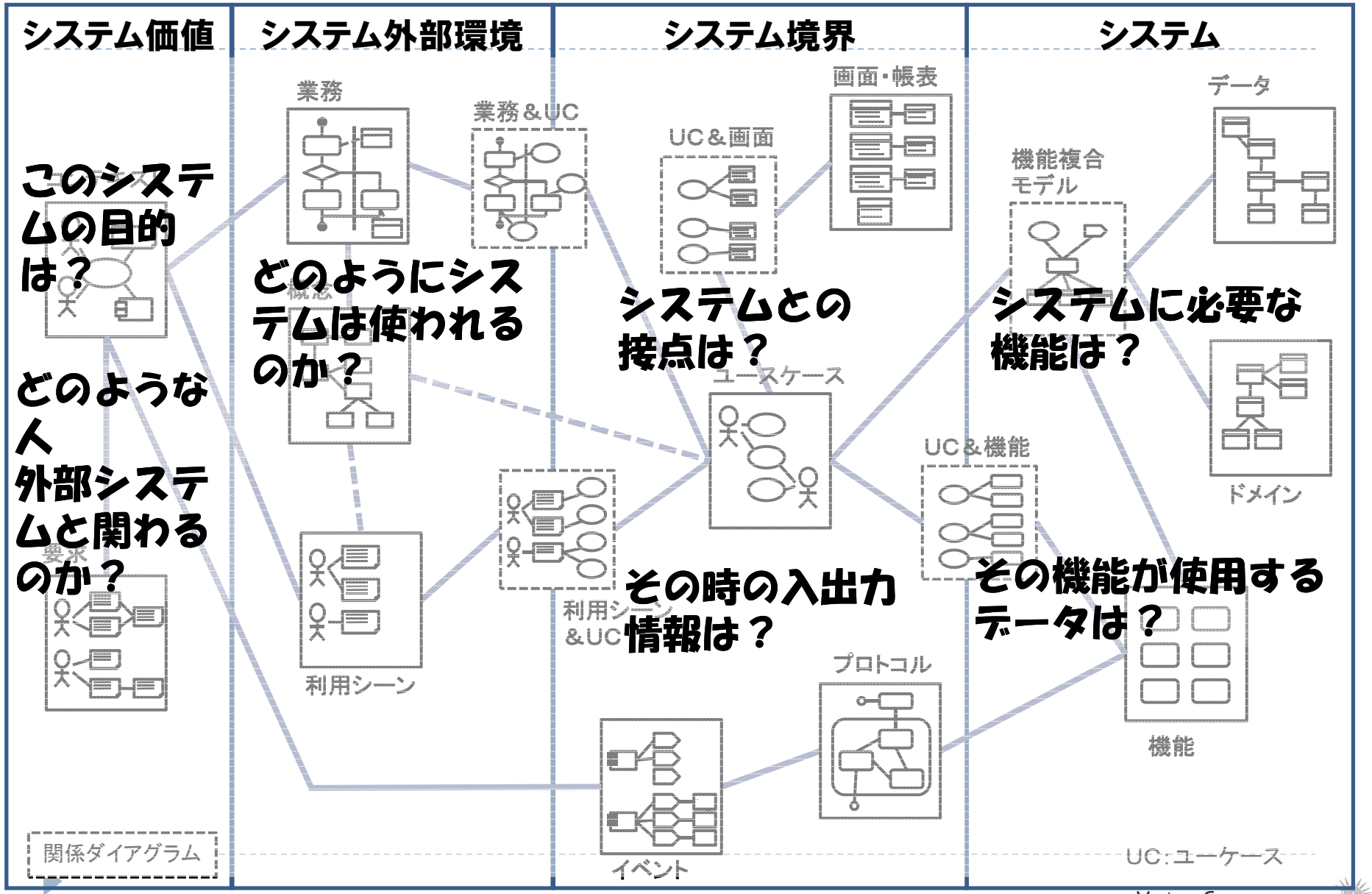
4.【システム】を定義する

- 機能を明らかにする
- データを明らかにする
- ドメインの構造を把握する

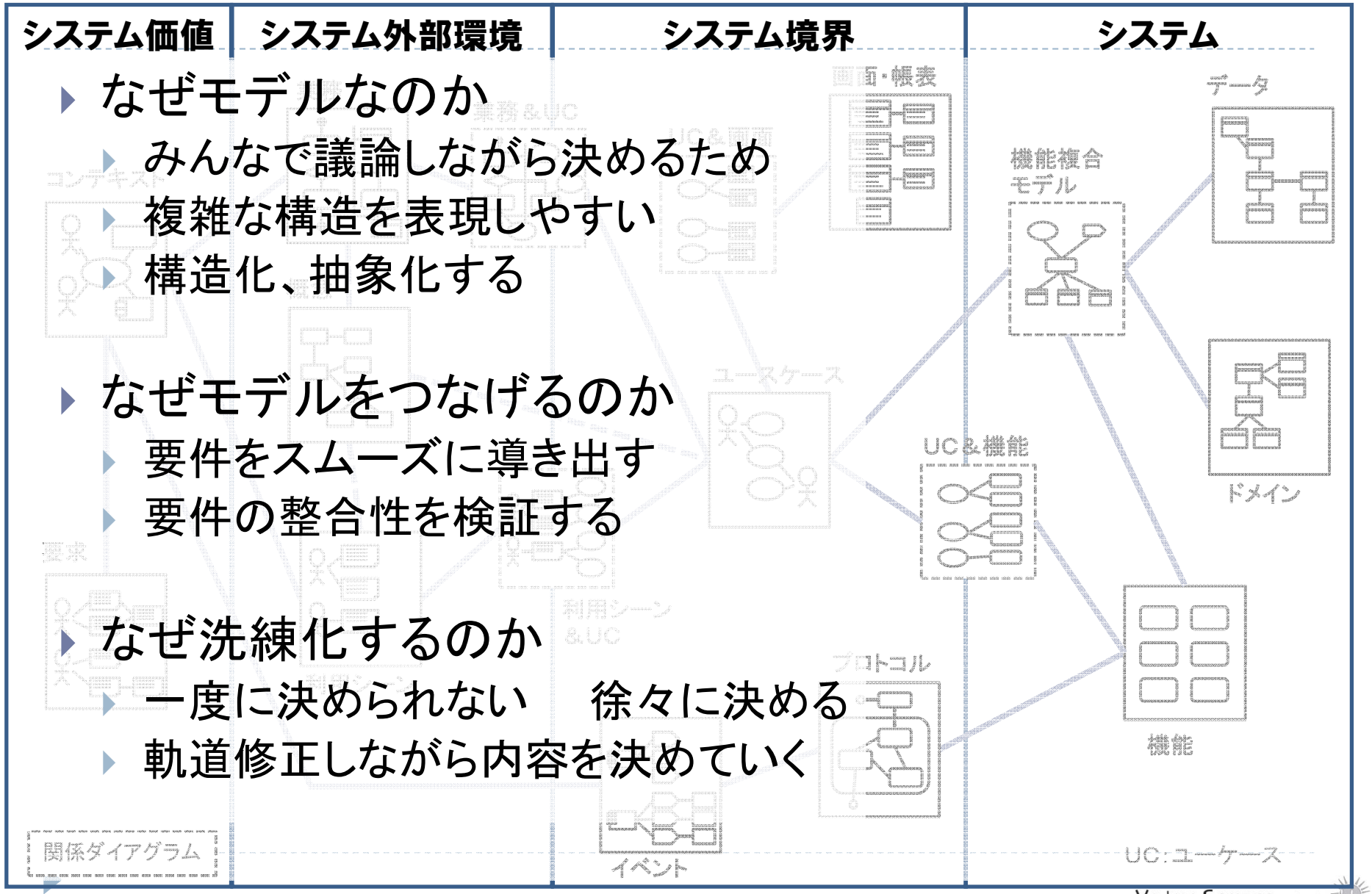
要件の構造にモデルを当てはめる



結局モデルで何を表現しているのか



なぜUMLを使うのか



要件の精度を高める

ダイアグラムの検証ポイント

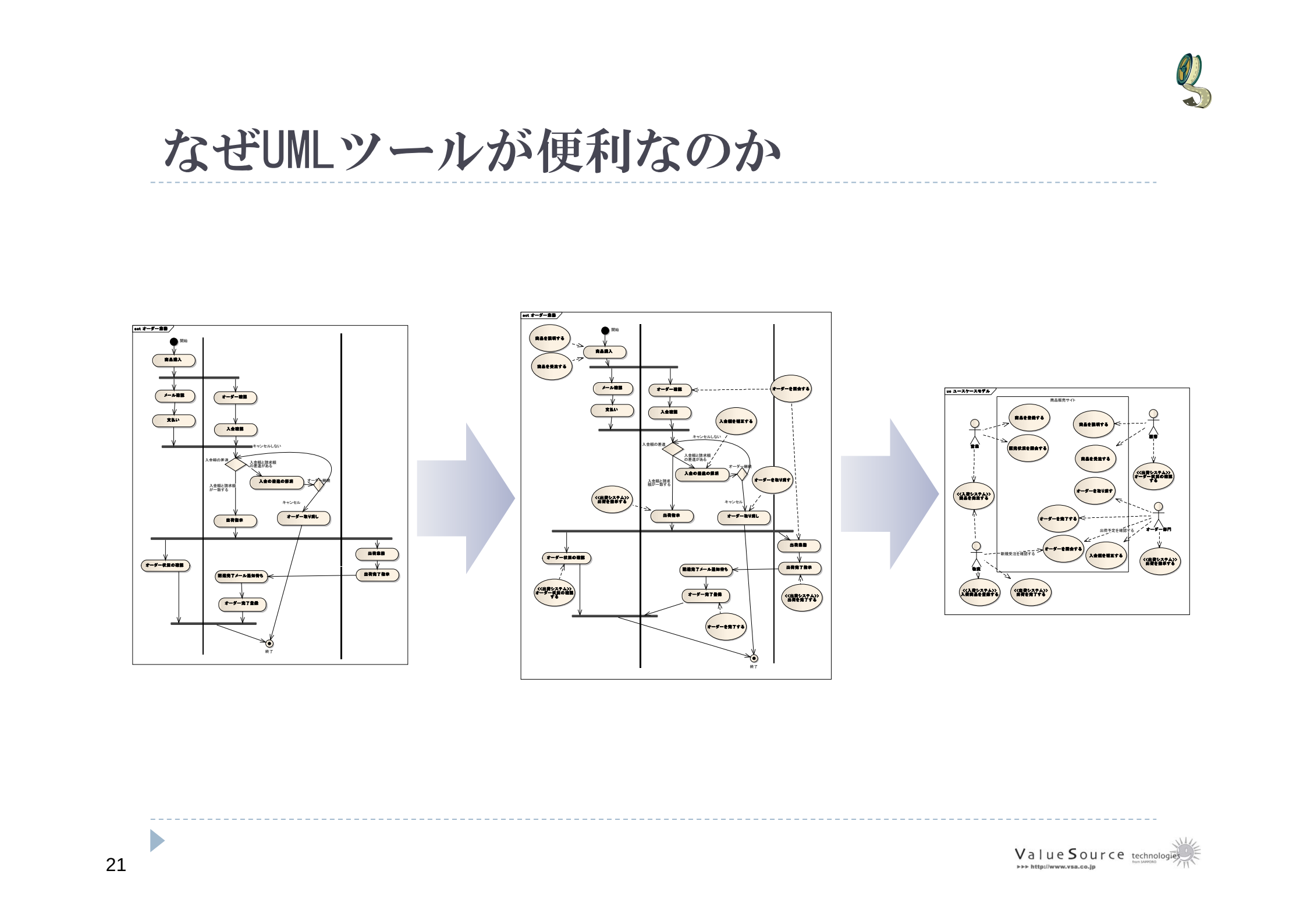
問い		検証内容
Q101 要求の元となる関係者にはどのような人がいるか	コンテキスト	□背景 要件定義を網羅的に進めるためにその出発点となる関係者を洗い出す必要がある □確認ポイント ・全ての関係者がアクターとして洗い出されているか ・システムに関わらないが業務に関わるアクターも含まれているか ・アクター同士の関係が示されているか ・アクターは役割の名前がついているか ・アクターの責務が明確になっているか
Q102 どのような外部システムと連携するのか	コンテ	□背景 外部システムとの連携もシステム化範囲を決めるための入り口になり、外部インターフェースの策定の出発点になる □確認ポイント
Q10 どの るか が		

整合性をあわせるポイント

問い	ダイアグラム	懸念事項
利用シーンには必ずユースケースが結びついているか	利用シーン	利用シーンに結びつくユースケースがない場合はユースケースが抜けている可能性がある。
利用シーンには必ずアクターが結びついている	利用シーン	利用シーンに結びつくアクターが無い場合は必要なアクターが抜けている可能性がある
ユースケースには必ずアクティビティが結びついているか	ユースケース	ユースケースに関わるアクティビティがない場合は必要なアクティビティが抜けている可能性がある。
ユースケースには必ず利用シーンが結びついているか	ユースケース	ユースケースに関わる利用シーンがない場合は必要な利用シーンが抜けている可能性がある。
ユースケースに結びついていない画面は無いか	ユースケース	ユースケースは画面か帳表、もしくは機能、データのどれかと結びついているはず。

ツールの支援

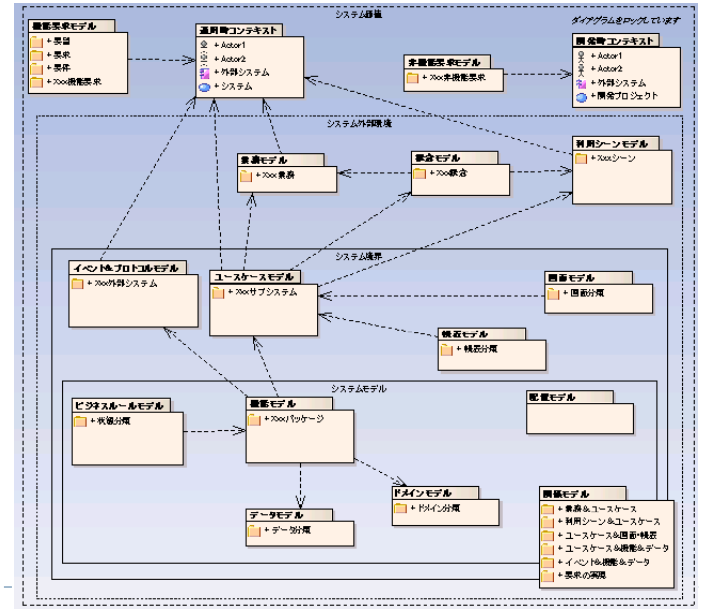
要件定義を行うために必要なツールや、要件を整理するたのテンプレート、並びに作成して要件定義のチェックツールを紹介します。

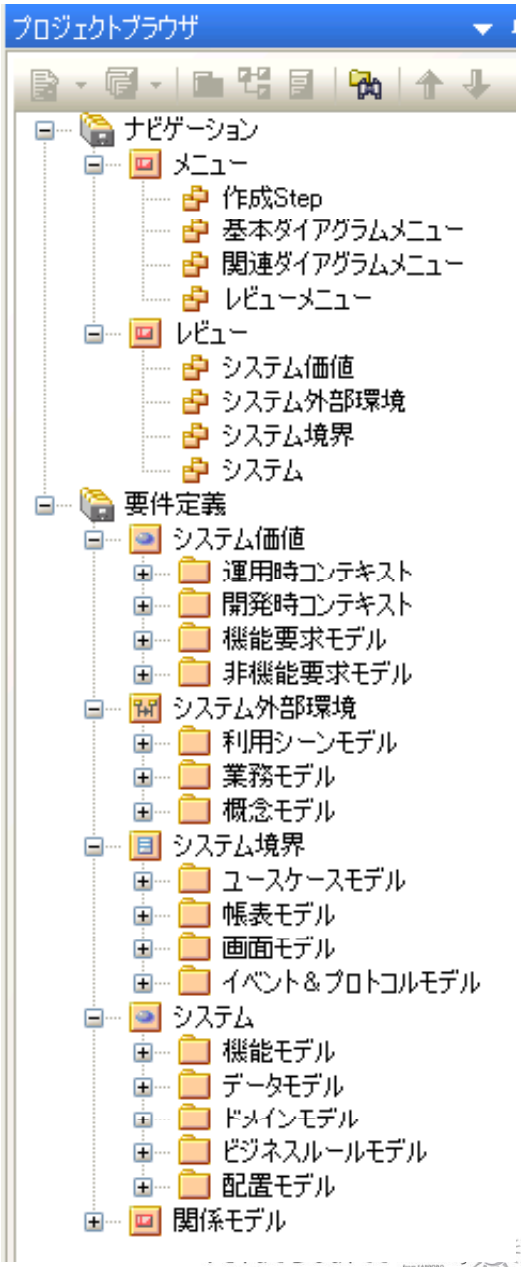


なぜUMLツールが便利なのか

RDRAテンプレート

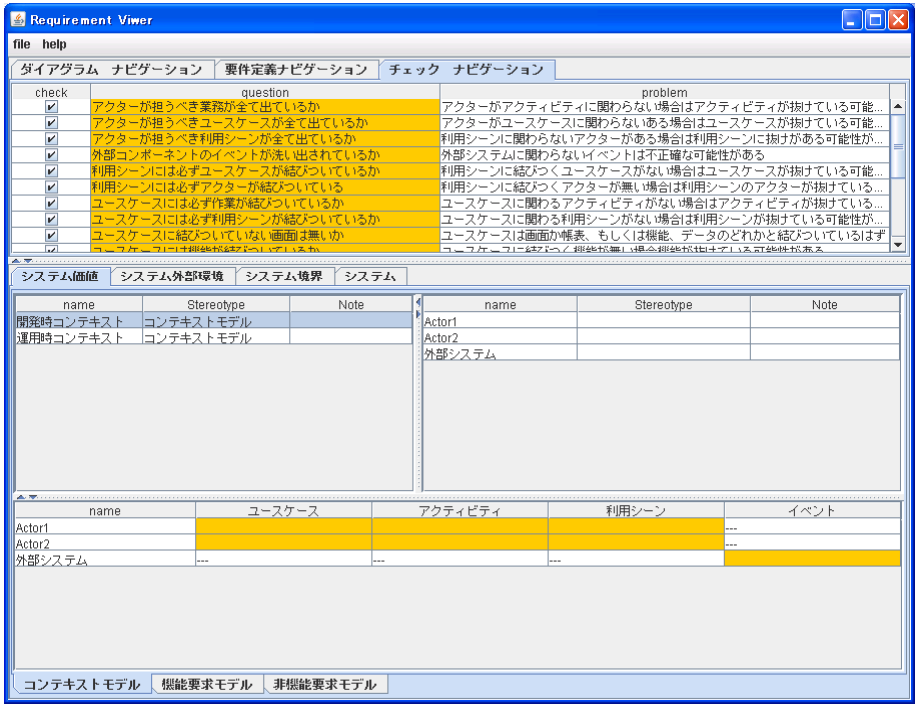
- RDRAで作成する一連のモデルをあらかじめパッケージとして分類している
- メニューが用意され必要なダイアグラムに直ぐに飛ぶことができる
- レビュー用のダイアグラムも用意されている





RDRACheker プログラム

- EAで作成したモデルの関係をチェックする
- 厳密なチェックではなくあくまで関係性を確認しモデルを見直す



EAを使うことで要件定義そのものを情報として扱える

要件分析プロセス

要件定義をどのように進めるのか

スケジュール管理と言えばガントチャート

でも上流工程ではガントチャートはうまく機能しないことが多い

混乱する上流工程

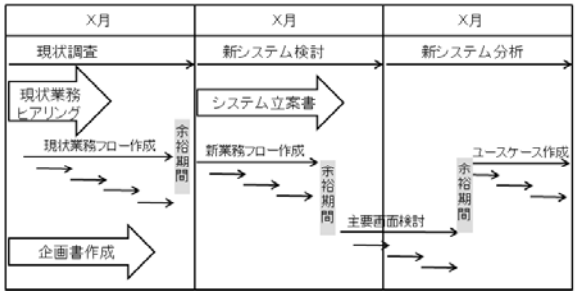
- あなたならどう答えますか？
  - **なぜ上流工程のスケジュール作成は難しいのか**
  - **なぜスケジュール通りに進まないのか**
  - **そもそも上流工程のスケジュール管理をどのように行えばいいのか**

- そのヒントをガントチャートから考えてみましょう
  - **スケジュール管理と言えばガントチャート**
  - **ガントチャートはどのような前提によって組み立てられるのか**

ガントチャートがイメージさせるもの

- ガントチャートを作成する（利用する）ときには以下のような考え方が背景にある
  - タスク間には依存関係があり依存されているものから先に手をつける
  - タスク間の依存関係に基づいてタスクを結びつけると完了日が明確になる
  - タスクの平行性を高めると期間を短縮できる
  - 「タスクを実現すれば想定している成果物を作成出来る」と考える
  - 担当者は割り当てられたタスクだけをこなせばいい
  - 全てのタスクは洗い出せる（出来ない時は猶予期間を用意する）
  - タスクは全て計画可能であり計画通りに実現可能であると考える

- ガントチャートがうまく機能するための条件
  - タスクへのブレークダウンが適切にできる
  - ほぼ予定された時間内にタスクが終わる



- 上記の条件が満たされるとき　ガントチャートはとても良くできた管理法である

- しかし、上記の条件が満たされないときはガントチャートはうまく機能しない

じゃあどうする？　⇒　時間を軸足にする

- スケジュールの3つパラメータ
  - 人：組織、リソース
  - 事：成果物、タスク
  - 時間

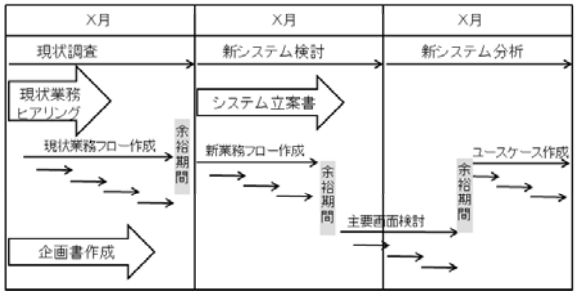
- 変わらない軸として時間を用いる
  - 事と人は変化が激しい　　時間は変わらない
  - 一定期間をタイムボックスとして扱う
  - サイクルを重視する
    - 作業にリズムを持たせて進行をスムーズにする
    - サイクルの中で作業方法を見直す

- 何が違うのか
  - ガントチャートを使って間に合うことを確認した方が安心だ！！
  - ⇒　確かに！！

  - 実体：副作用が深刻な問題を引き起こす
  - ⇒　実体とかけ離れたスケジュールで意味のない進捗管理を行う

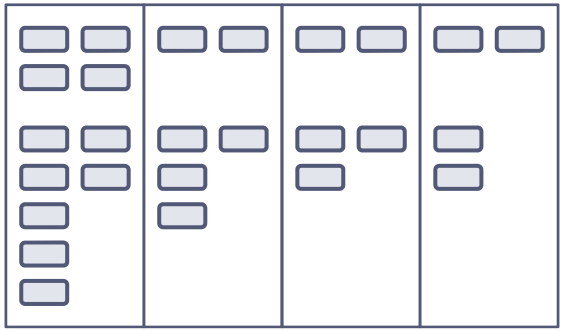
  - **最初に決めてしまわない**

ガントチャート



事と人で組みたてる

タイムボックス



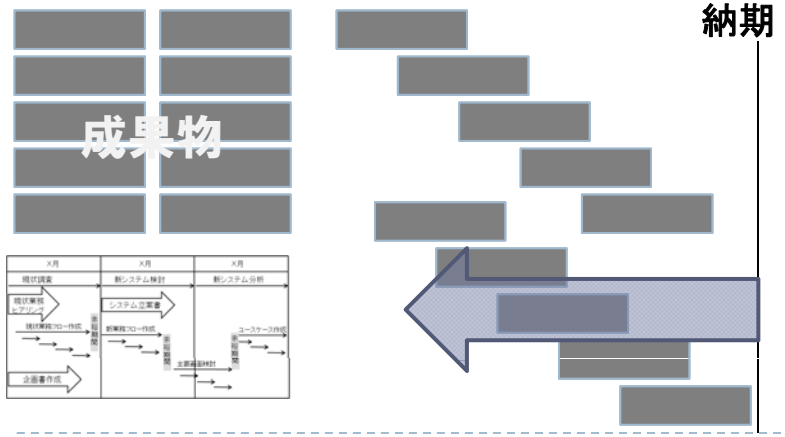
直近の時間枠に事と人を当てはめる

考え方を変える　成果物中心から時間中心

- 従来:
  - 成果物を固定化してそれを決められた期間の中で実現する
- RDRAプロセス:
  - タイムボックス毎に作業の方法を変えて必要なものを作成する
- 軌道修正が必要な要件定義工程では成果物を固定化する方式はそぐわない

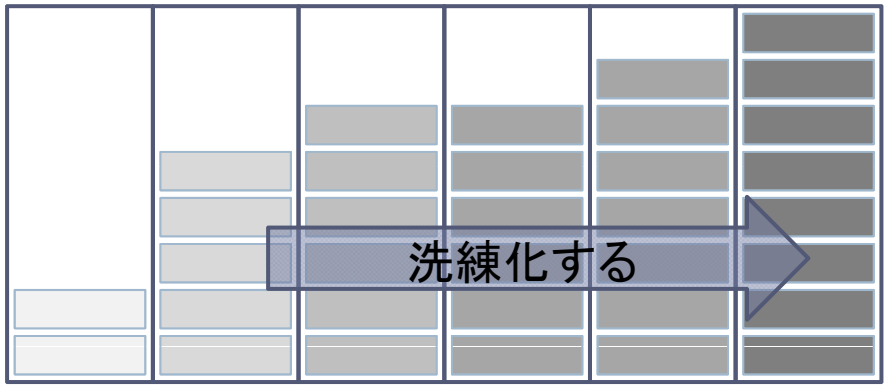
ガントチャートベース

成果物は変えずに納期に合わせて平行性を高めて納期に納める



タイムボックスベース

タイムボックス毎に作業の内容や成果物の粒度、精度を調整し洗練化しながら最低限必要なものを作成する



何を捨て　何を重視する

- 捨てるもの
  - 「スケジュールを安易に変えることは悪いことだ」という考え
  - 「やることが全て見えている」という幻想
  - 「うまくやれば手戻りはなくせる」という幻想

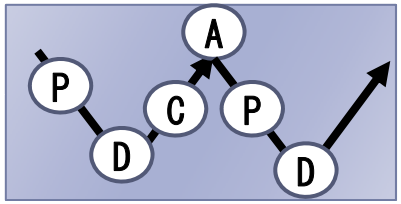
- ラフに決め　直近の作業を柔軟に決めていく
  - そもそもパラメータ（人とタスク）の変化が大きいところでは、成果物の作成量を最大にするのではなく、変化に適切に対応することを目標とする方が効果的である

- つまり
  - 不正確なガントチャートに縛られて不適切な作業をするよりは、先々までの詳細なスケジュールが見通せないけども、適切に軌道修正しながら進める方が有効であると考える

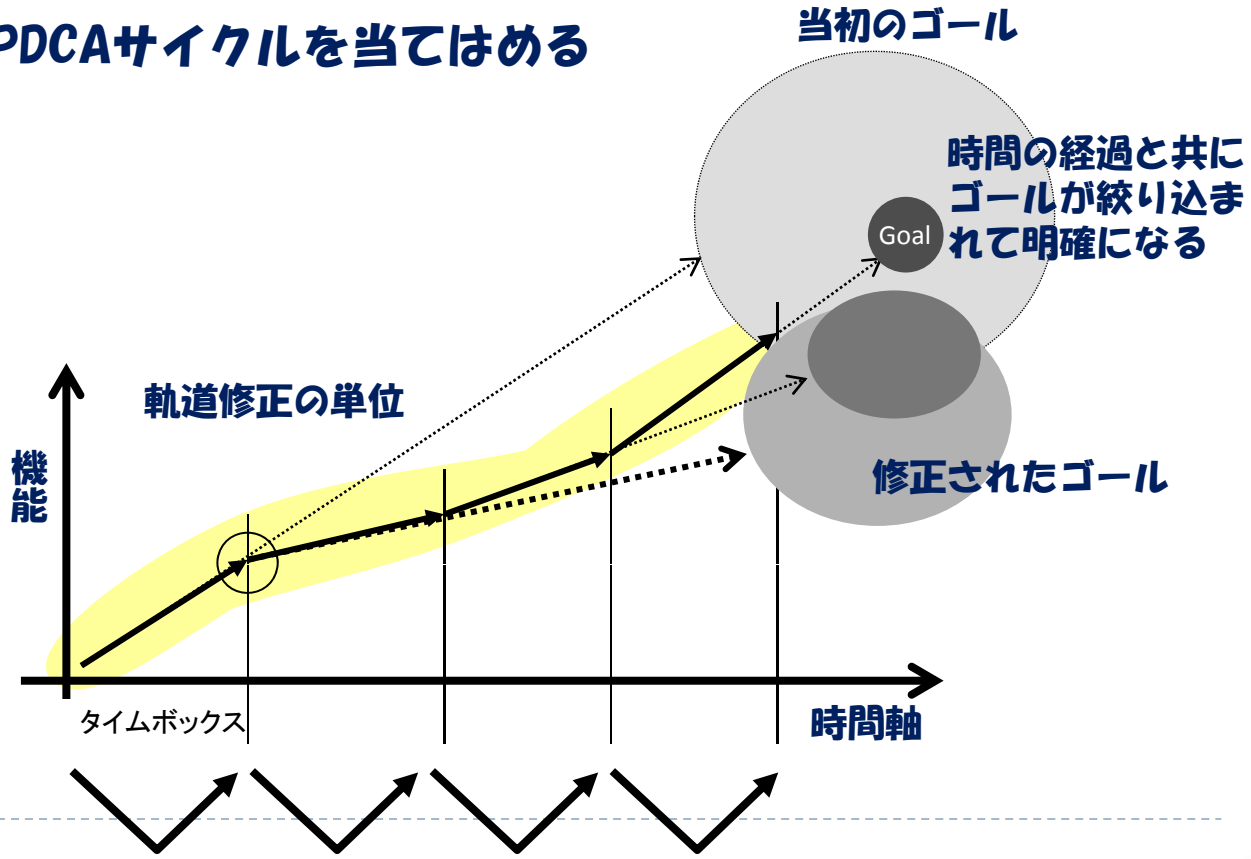
- 結局
  - タスク（成果物）中心であろうが、時間中心であろうが3ヶ月先の詳細なスケジュールなど誰も組めない
  - それを受け入れていかにスケジュール管理を行うかを考える

ゴールについて

- ゴールは時間とともに明らかになる
- ゴール目指して軌道修正が必要
- タイムボックスで軌道修正する
- １タイムボックスにPDCAサイクルを当てはめる



V字モデル

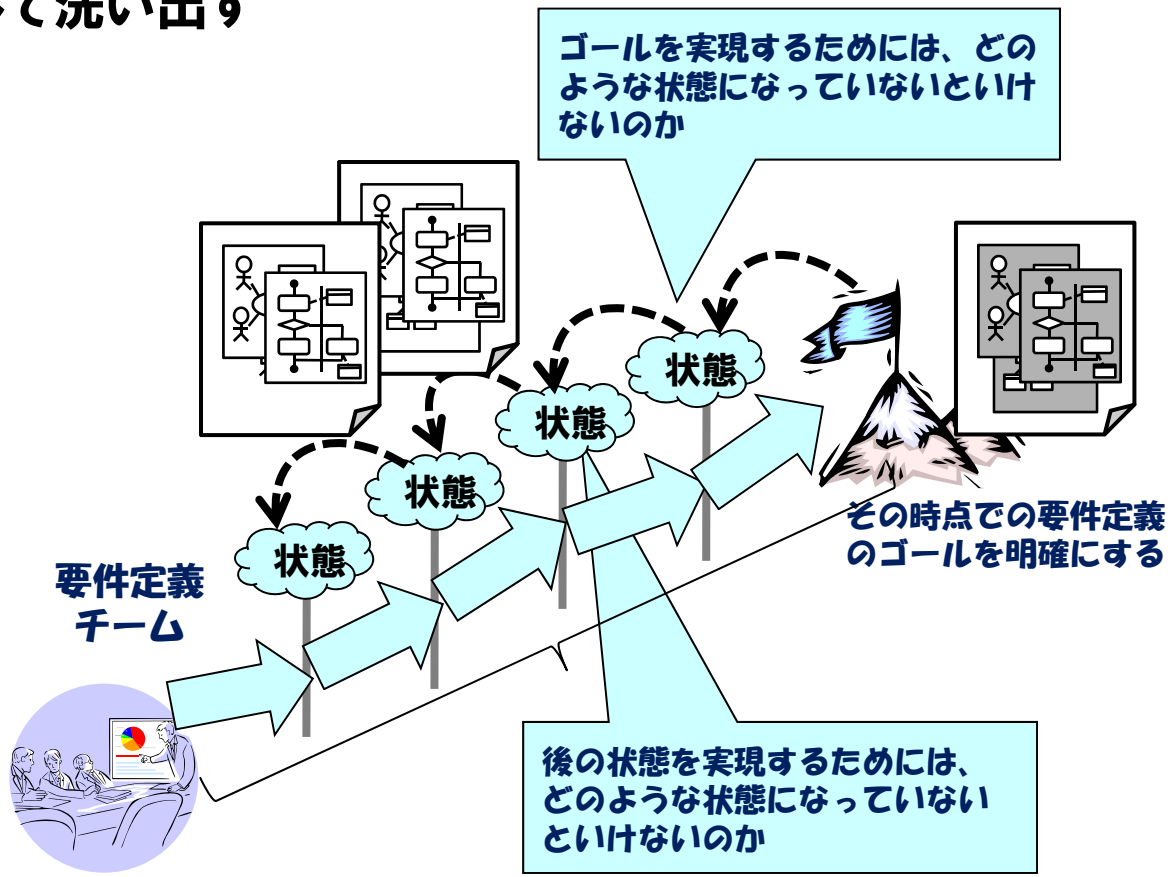


マイルストーンは状態として管理する

要件定義のゴールを決め、そのゴールから遡って
各マイルストーンを状態として洗い出す

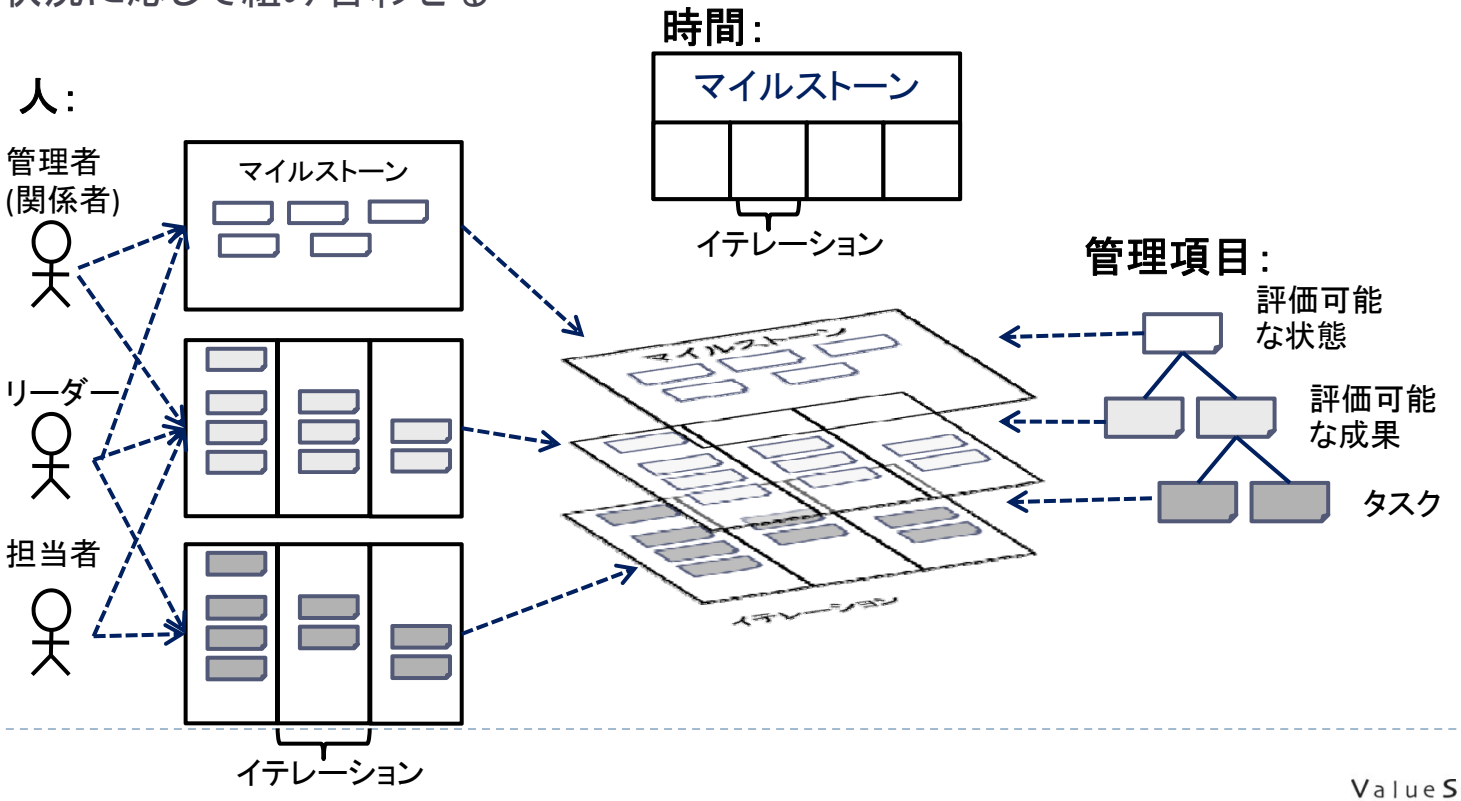
マイルストーン候補

フェーズ	マイルストーン
状況把握	自己理解度を測る
	現状分析
方向性の確立と組立	コア把握と方向性の確立
	システム化への組立
	網羅性の確保
整合性確保	整合性確保
	項目レベルで整合性確保



繰り返しをどのように管理するか

- 複数視点に対応するスケジュール管理
  - 階層化した管理項目
    - 人　　管理者、リーダー、担当者
    - 時間　　マイルストーン、イテレーション
    - 管理項目　　状態、成果物、タスク
  - 状況に応じて組み合わせる



まとめ

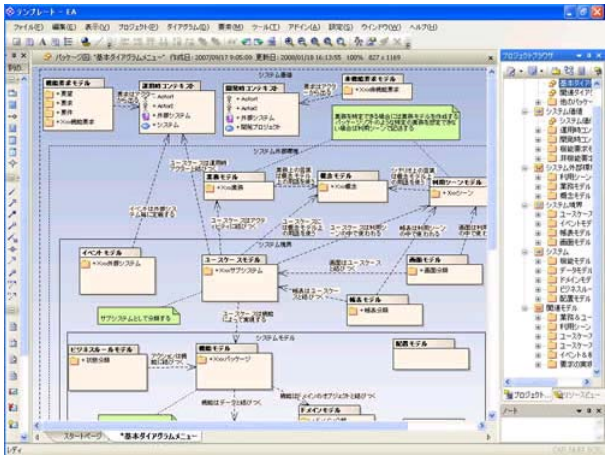
- 要件分析フレームワーク
  - 要件定義の枠組みを利用する
    - システム価値、システム外部環境、システム境界、システム
  - 上記の枠組みに適したモデルを使って要件を定義する
  - 要件定義の各情報はつながっており、そのつながりを利用することでシステマティックに精度の高い要件定義が可能となる
  - それを何度も繰り返して要件を洗練化する
    - 要件定義の構造から網羅性や整合性を確認するポイントが出てくる
    - 一つ一つの成果物を完成させない

- 要件分析プロセス
  - 「タスク」「人」が安定しないのであれば、安定している「時」をベースに管理し、状況の変化に対応する方が有利である
  - タスク、時間、人を階層化し複数のレベルで管理する
  - タイムボックス毎に軌道修正しながらゴールを明らかにし、納期に納める
  - 要件分析フレームワークの一連のモデルの作成、洗練化を繰り返し精度を上げる

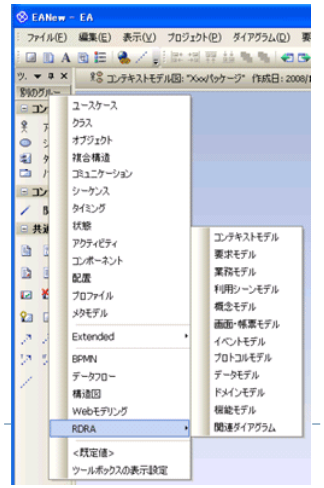
リレーションシップ駆動要件分析の情報

http://k-method.jp

RDRA用テンプレート



RDRA用プロファイル



書籍

＜リレーションシップ駆動要件分析の情報サイト＞
UMLツールのEnterpriseArchtectのテンプレート
やプロファイルがダウンロード出来ます