品質を創造する

VERISERVE

JaSST'23 Tokyo

ソフトウェアテストシンポジウム 2023 東京

# MBTで目指すこれからのテストづくり

2023年　3 月　9日

講演資料

株式会社ベリサーブ

研究企画開発部　技術戦略課

須原秀敏　　蛭田恭章

# 本講演について

VERISERVE

1. モデルベースドテスト（MBT）概要
2. 当社の考えるMBT
3. MBT事例紹介
4. MBTツールの紹介

# 須原　秀敏（すはら　ひでとし）



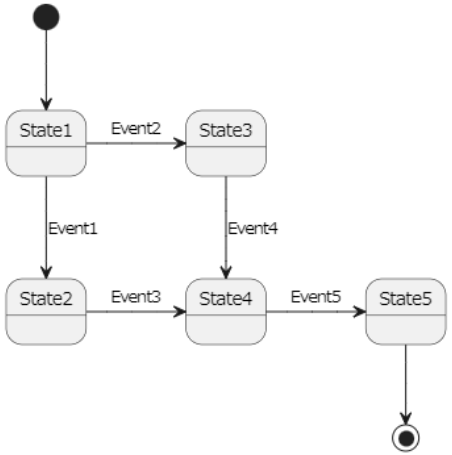

- 2010年 株式会社ベリサーブ 入社
- 主に車載関連のソフトウェア/システムテスト業務に関わる
- 2021年から研究企画開発部にて、モデルベースドテストやAIの品質保証技術の開発に従事
- 2022年からベリサーブベトナムにて研究開発組織の構築に従事
- 社外活動：JSTQB技術委員、JTC1/SC7/WG26エキスパート
- Twitter: @suhahide

# 1. モデルベースドテスト（MBT）概要

# MBTとは

VERISERVE

- **テスト設計を自動化する技術**
- **モデル（形式化された仕様）を入力して、決められた変換ロジックを使って、テストケースを生成する**
  - 状態遷移テスト（Nスイッチ）やデシジョンテーブルテストはMBTの代表格


「連続するN+1個の遷移を網羅するテストケースを生成する」

# MBTの一般的なメリット

VERISERVE

- **説明性**
  - 自プロジェクト内でMBTパターンを作る場合でも、どうやってそのテストケースが生まれたかを説明できることは価値がある
- **トレーサビリティ**
  - MBTモデル（場合によっては開発モデル）からテストケースへのトレースが必然的に取れる
- **テスト設計の品質向上**
  - モデルを描ければ、誰でも同じテスト設計が再現できる
    - 特に同じプロジェクトであれば、テスト戦略が同じであることが多いため効果が高い
  - MBTパターンの追加により過去の欠陥を再発させないためのテスト設計のカイゼンが実現できる
- **メンテナンスコスト**
  - 単一のメンテナンスポイント
- **アジリティ**

# 2. 当社の考えるMBT

# ベリサーブはMBTやります！

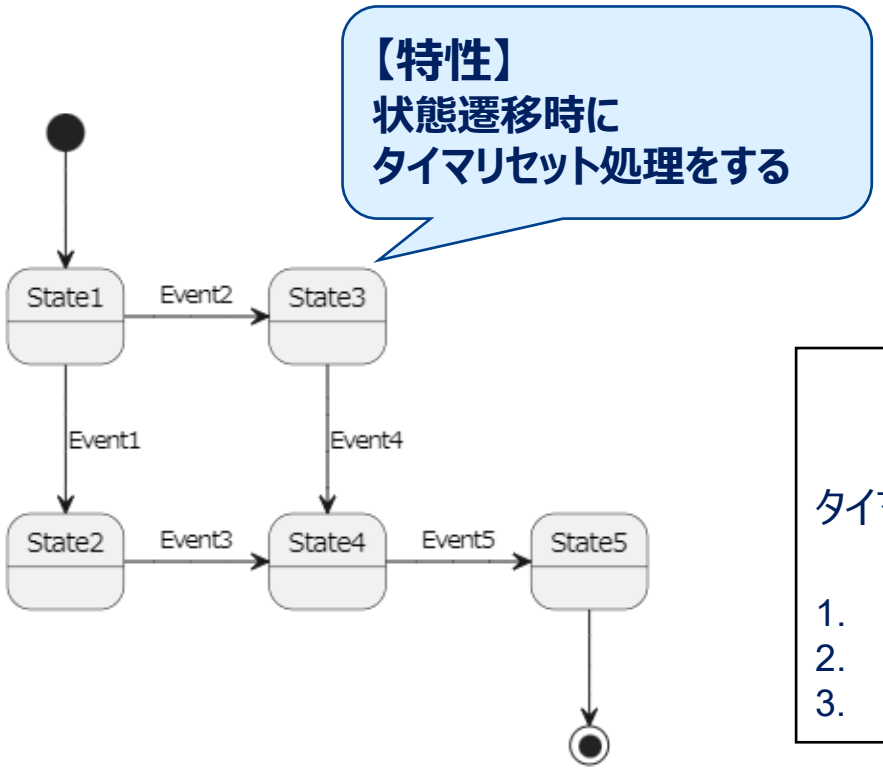
- なぜMBTに注力するか、理由を後述します

# ベリサーブにとっての意義

VERISERVE

- **MBTパターンを蓄積して再利用することで、テスト設計の知恵が「ちゃんと」たまる**
  - 職人技から会社のチカラに
  - 個人プレイから組織プレイに
- **別の言葉で言うと、MBTによりテストプロセスをデジタル化し、コアなデータを再利用可能にする**

**MBTパターン（ベリサーブ用語）：**
**開発モデル（仕様）上のある特性に対して、このような狙いのテストを生成するという一連のロジック**



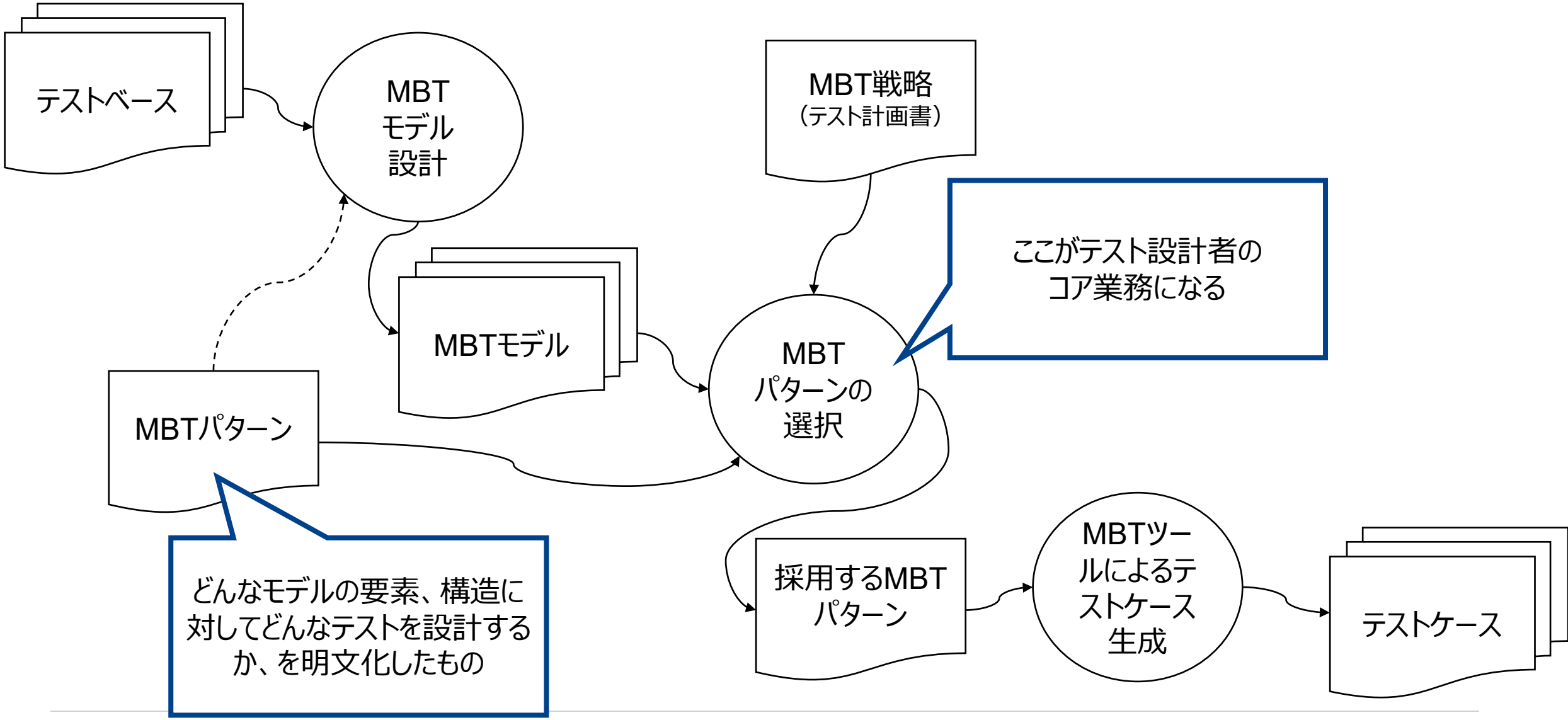

テストケース生成ルール

タイマリセットがある場合以下を確認するテストケースを生成する

1. 遷移条件の時間経過の半分を経過時(Tとする）に遷移元に戻る
2. 遷移元の状態で、T経過時に、対象状態からの遷移先に遷移しないこと
3. 対象状態に遷移後、遷移条件の時間経過後（2*T)に、遷移先に遷移すること

# 当社が考えるMBTのプロセス（超概要）

VERISERVE



# 当社の既存ツールとの関係：GIHOZ

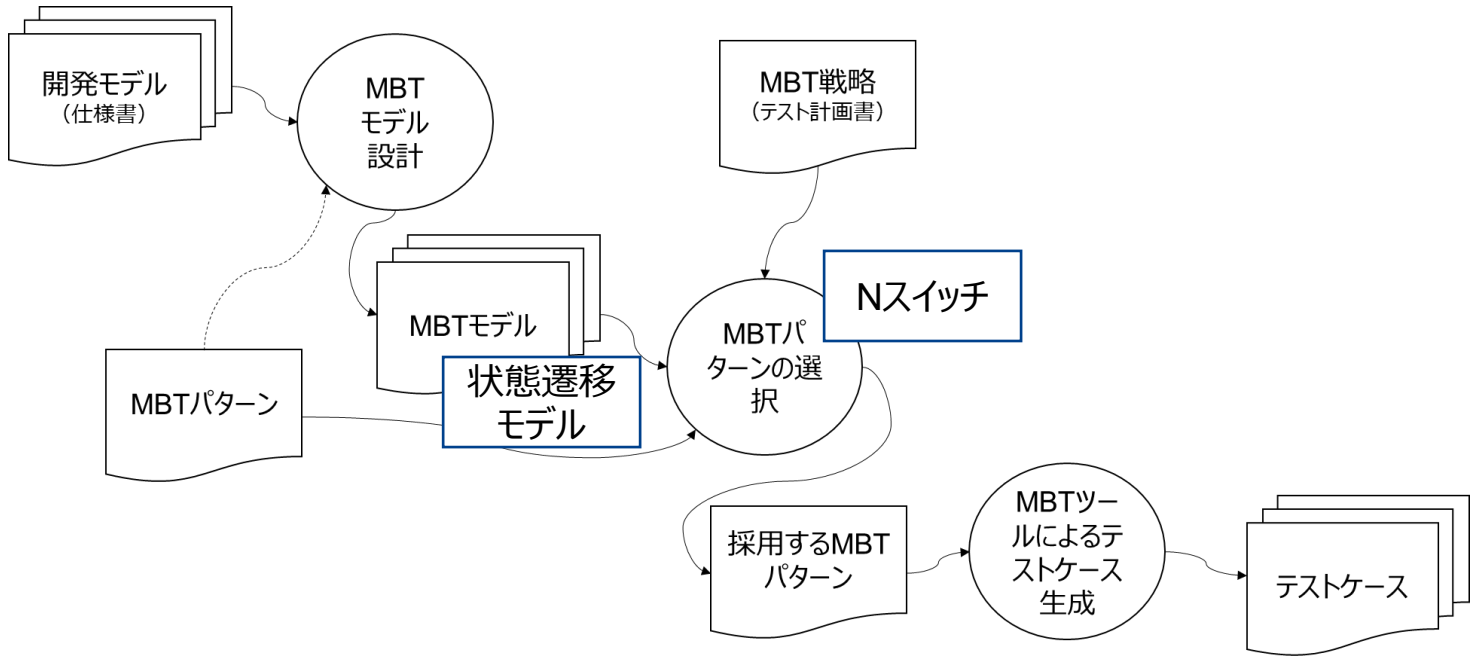
VERISERVE

## GIHOZとは

- テスト技法をブラウザベースで使えるクラウドサービス

## GIHOZとの関係性

- GIHOZは、MBTモデルを設計し、あらかじめ定められたMBTパターンを適用し、テストケース生成をするツール
- GIHOZは、限定された（そして非常によく使われる）MBTパターンを使いやすくするツール
- すごく簡単に言うと、ベリサーブが目指すMBTは、GIHOZが実現していることをもっと広範囲で対応できるようにすること




GIHOZ

# 当社の既存ツールとの関係：TREES（テスト観点DB）

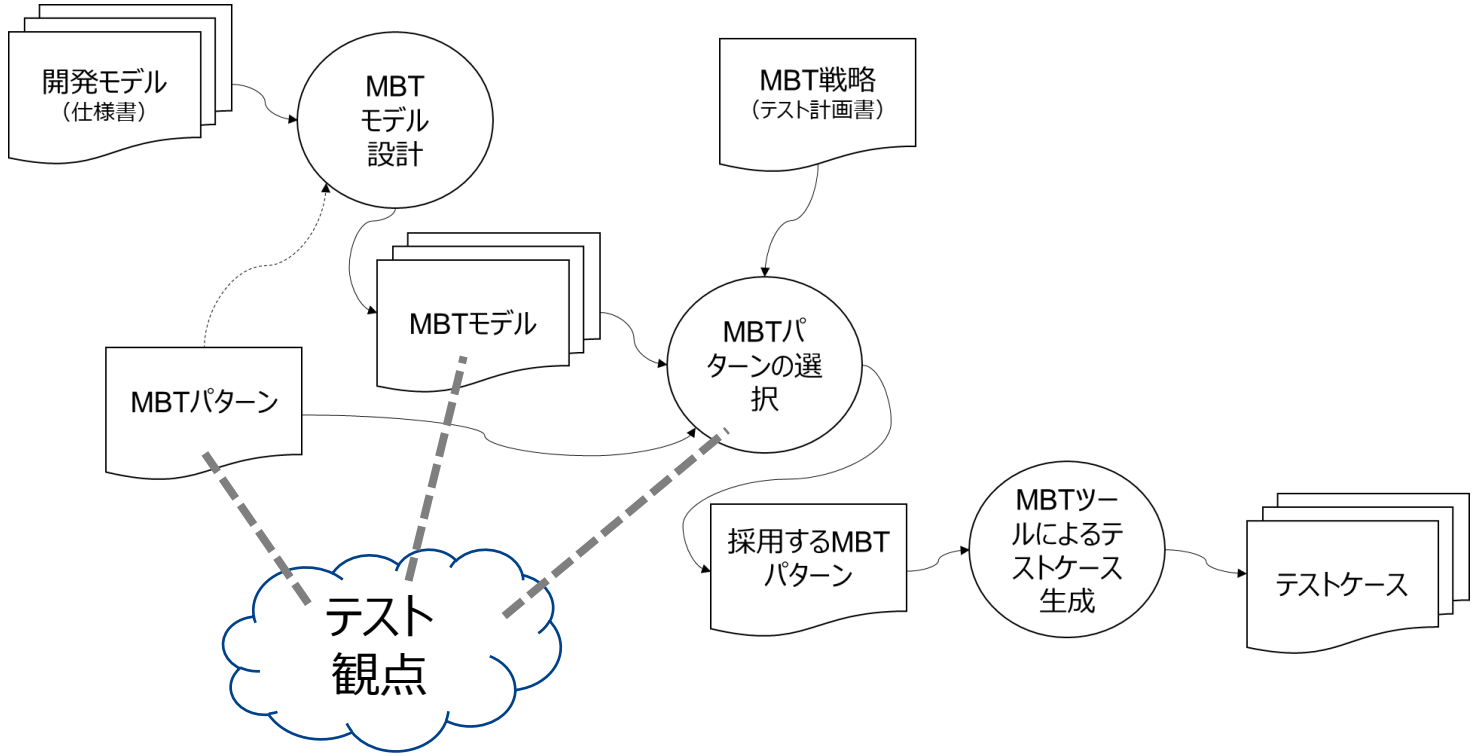
VERISERVE

## TREESとは

- テスト観点を蓄積する当社社内サービス

## テスト観点との関係性

- テスト観点は以下を含む
  - MBTモデルの要素や構造の特徴
  - MBTパターンのテスト生成ロジック
  - MBTパターンごとのカバレッジ
- すごく乱暴に言うと、テスト観点はMBTパターンのレベルまで形式化できなかった暗黙知の集合
- テスト設計者に任されていたテスト観点の適用を、より広いレベルのテスト設計者に行ってもらいたい

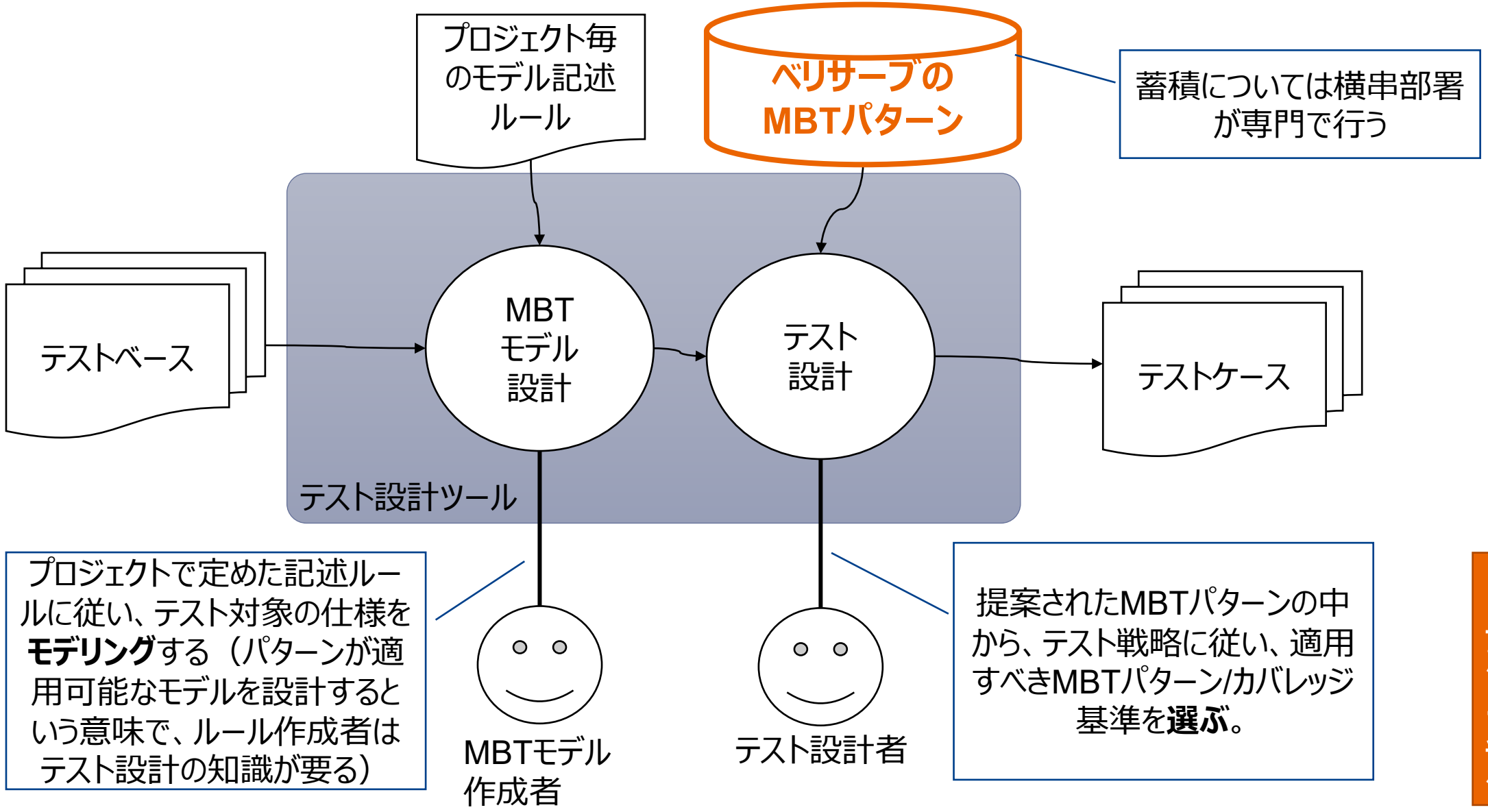



テスト観点DB

TREES

# 当社が提供するTester's Experience

VERISERVE




つくる から 選ぶ へ

# 蛭田　恭章（ひるた　やすあき）

VERISERVE

- 2006年 株式会社ベリサーブ 入社
- 家電やスマートフォン等の組み込み機器の
  ソフトウェアテストやソフトウェア品質全般の業務に携わる
- 2021年から研究企画開発部にて、新たな技術調査や技術開発に従事
- 現在は、モデルベースドテストやユーザビリティ/UX評価の研究開発に取り組む

# 3. MBT事例紹介

# ベリサーブでのMBT実践事例

VERISERVE

## ➢ 車載サプライヤ

- 対象モデル：アクティビティ図、ステートマシン図、ブロック定義図

## ➢ 自動車メーカー

- 対象モデル：ステートマシン図

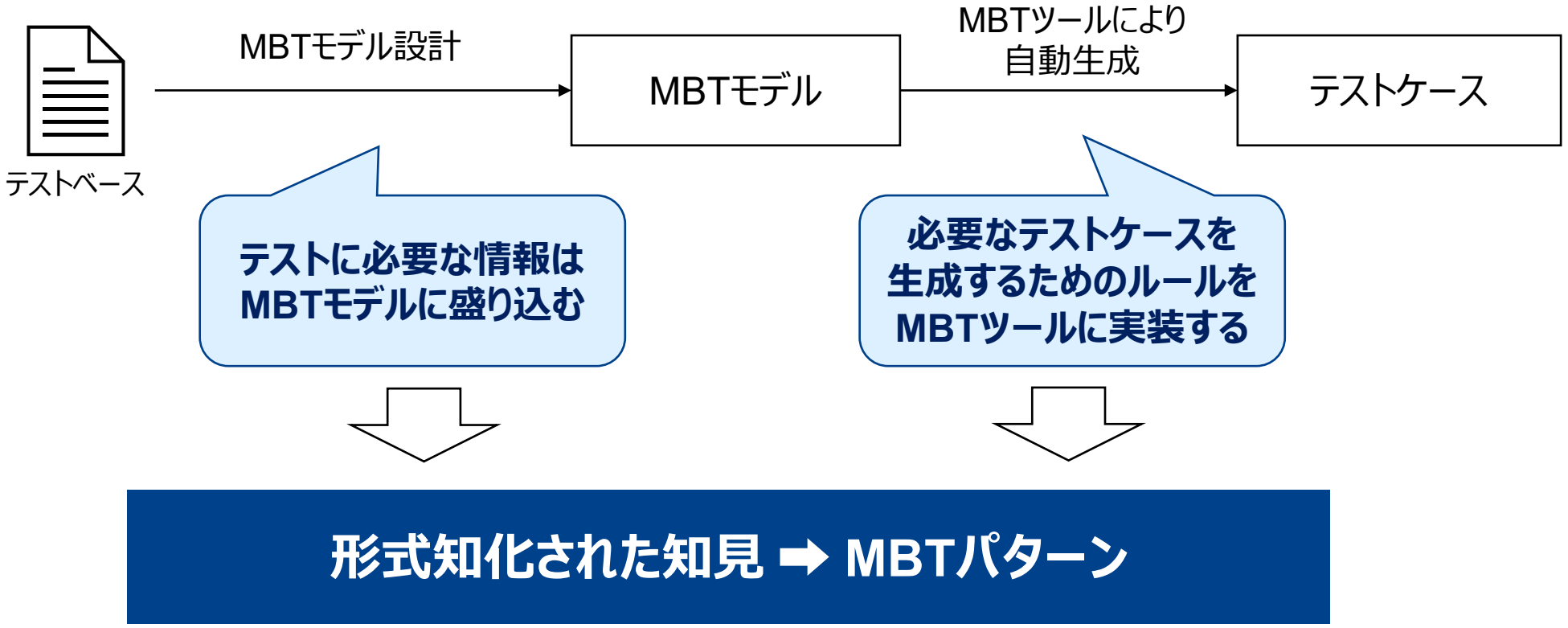
## ➢ モバイルアプリ（プラットフォーム）

- 対象モデル：WebAPI仕様モデル(Swagger)、デシジョンテーブル

# MBTをする上で大事なポイント

VERISERVE

- **MBTモデル設計やテストケース生成に必要な知見を、再利用可能な状態で形式知化することが大事**



# 知見を形式知化するアプローチ

VERISERVE

## ➢ テスト設計者が持つ知見

- 例えば、熟練したテスト設計者は**暗黙的な仕様**を捉えることができる
- 作成したテストケースをリバースモデリングすることでMBTで適用可能なように形式知化する

→ **WebAPIテストでの事例を紹介**

## ➢ 過去不具合から得た知見

- 例えば、**重大な不具合**の一部にはテスト観点として漏れていたものがある
- 不具合の発生条件をモデリングしMBTで適用可能なように形式知化する

→ **状態遷移テストでの事例を紹介**

# 知見を形式知化するアプローチ



## ➢ テスト設計者が持つ知見

- 例えば、熟練したテスト設計者は**暗黙的な仕様**を捉えることができる
- 作成したテストケースをリバースモデリングすることでMBTで適用可能なように形式知化する

**WebAPIテストでの事例を紹介**

## ➢ 過去不具合から得た知見

- 例えば、重大な不具合の一部にはテスト観点として漏れていたものがある
- 不具合の発生条件をモデリングしMBTで適用可能なように形式知化する



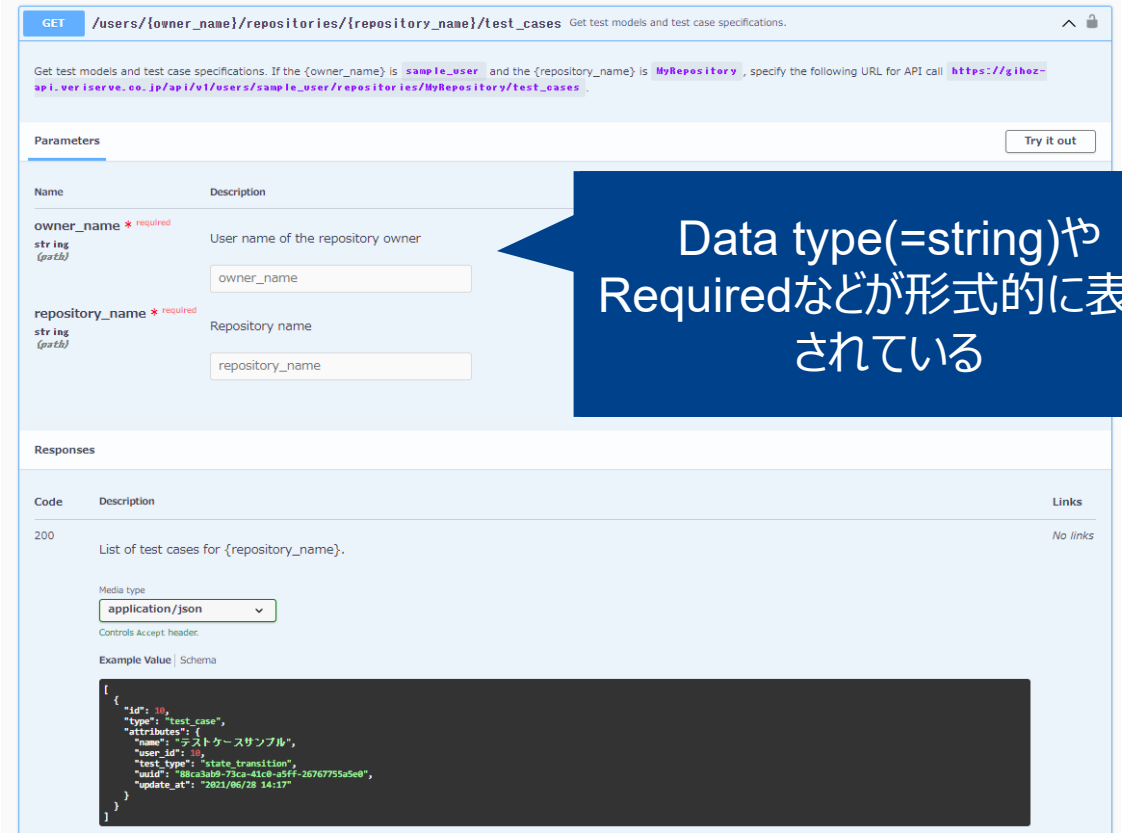
**状態遷移テストでの事例を紹介**

# WebAPI仕様のフォーマット

VERISERVE

- **今回の取り組みの全てのプロジェクトでWebAPI仕様はSwaggerにより形式的に表現されていた**
  - Swaggerは「Open API Initiative」という団体が推進しているRESTful APIのインターフェイスの記述をするための標準フォーマット
  - https://swagger.io/specification/

(Swagger UI)




Data type(=string)やRequiredなどが形式的に表現されている

(swagger.yaml)

```yaml
/users/{owner_name}/repositories/{repository_name}/test_cases:
  get:
    tags:
    - "TestCases"
    summary: "Get test models and test case specifications."
    description: "Get test models and test case specifications.
      If the {owner_name} is `sample_user` and the {repository_
      `https://gihoz-api.veriserve.co.jp/api/v1/users/sample_use
    parameters:
    - name: "owner_name"
      in: "path"
      schema:
        type: string
      description: "User name of the repository owner"
      required: true
    - name: "repository_name"
      in: "path"
      schema:
        type: string
      description: "Repository name"
      required: true
    responses:
      200:
        description: "List of test cases for {repository_name}."
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/TestCases'
              xml:
                name: "TestCases"
```

# 形式知化までの流れ

VERISERVE

## 1. 過去のテストケース（テストスイート）を収集する

- プロジェクト数：4
- WebAPIメソッド数： 19 (GET:10, POST:6, PUT:1, DELETE:2)
- テストケース数：268

## 2. テストケースの内容と入力となったWebAPIの開発仕様を確認する

## 3. Swaggerにて形式的に明記されているものを「明示的な仕様」、そうでないものを「暗黙的な仕様」に振り分ける

- 実際暗黙的な仕様に振り分けられた中には以下の仕様が存在した
  - Swagger以外の文書で仕様の補足が記載されている
  - 開発者とのQ&Aに記載されている

## 4. 「暗黙的な仕様」のうち共通性のあるものを暗黙的になりがちな仕様に共通するパターンとする

# 暗黙的な仕様とした例

VERISERVE

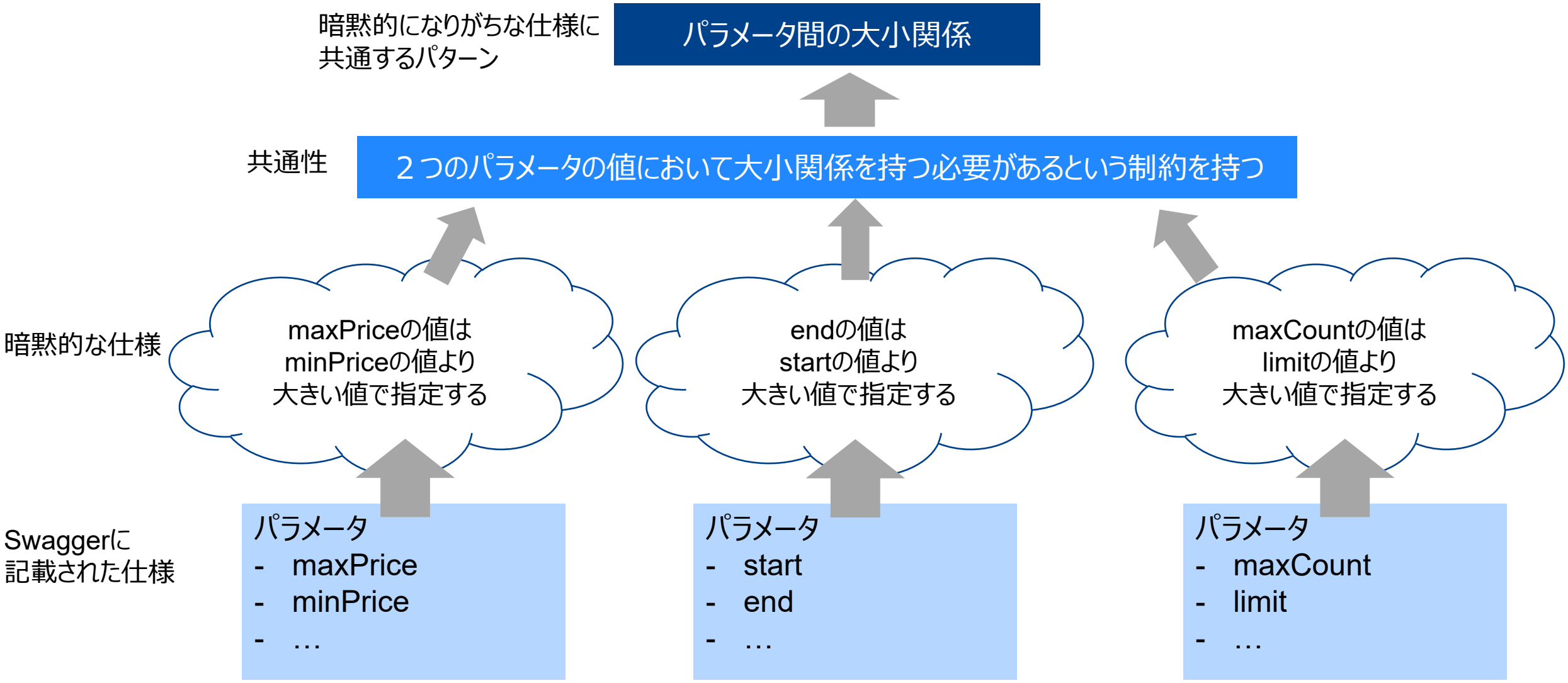
## 既存のテストケースを分析した結果、暗黙的な仕様とした例を以下に示す

- **パラメータ間に関係性があった例**
  - maxPriceはminPriceより大きい値をValueとして指定する必要あり
- **指定する値に制約があった例**
  - Request Bodyで指定するKey ”id” において、Request Body内で重複するidは指定不可
- **API利用の前提として条件があった例**
  - 該当API利用の際は以下のパーミッションが必要
    - PROFILE
    - USER_LIST

# 暗黙的な仕様の共通性

VERISERVE

## 以下の例にある通り、暗黙的な仕様で共通性のあるものをパターンとして蓄積した



# WebAPI仕様にて暗黙的になりがちな仕様に共通するパターン

VERISERVE

➢ **今回は以下の17パターンを蓄積した**　※以下ツリーの最下層の項目をカウント

**『前提』仕様のパターン**　6パターン

- API利用の前提条件
- パラメータ使用の前提条件
- APIのアクセス制限
  - 権限
  - リクエスト頻度
- API利用のコンテキスト
  - 過去バージョンのAPI有無
  - 利用者属性

**『関連』仕様のパターン**　11パターン

※以下ツリーで表現されたパターンの他に、仕様更新前後の関連として3パターン

- 関連を考慮した値指定
  - 複数値指定可能
  - (DBに)存在する値の要指定
  - 特定パラメータに依存した値指定
  - ユニークな値を要するKey
- パラメータ間の関連
  - パラメータの要同時指定
  - パラメータ間の親子関係
  - パラメータ間の排他関係
  - パラメータ間の大小関係

# 暗黙的な仕様をMBTモデルに盛り込んだ例

VERISERVE

**Swaggerの記載**
（イベントを更新するAPIのPUTメソッド）

```
event
 {
 event_id integer
 event_name string
 }
```

Request Bodyに指定するevent数を意識したMBTモデル

**明示的な仕様から設計したMBTモデル**

| | | | | | |
|---|---|---|---|---|---|
| | | 有効/無効 | ☑ | ☑ | ☑ |
| | | | 1 | 2 | 3 |
| 条件 | 指定するevent数 | なし | Y | N | N |
| | | 単一 | N | Y | N |
| | | 複数 | N | N | Y |
| 動作 | Success | | - | X | X |
| | Error | | X | - | - |

**「ユニークな値を要するKey」というパターンを活用し、暗黙的な仕様を調査する**

『event_id』はRequest Body内で重複した値の指定は不可という暗黙的な仕様（制約）を捉える

パターン活用により
event_idが重複する場合、しない場合を条件として追加する

**暗黙的な仕様を盛り込んだMBTモデル**

| | | | | | | |
|---|---|---|---|---|---|---|
| | | 有効/無効 | ☑ | ☑ | ☑ | ☑ |
| | | | 1 | 2 | 3 | 4 |
| 条件 | 指定するevent数 | なし | Y | N | N | N |
| | | 単一 | N | Y | N | N |
| | | 複数 | N | N | Y | Y |
| | event_idの値 | 重複なし | Y | Y | Y | N |
| | | 重複あり | N | N | N | Y |
| 動作 | Success | | - | X | X | - |
| | Error | | X | - | - | X |

新たに生成されるテストケース

# 暗黙的な仕様から新たなMBTモデルを設計した例

VERISERVE

**Swaggerの記載**
（指定した範囲のスコアの値を取得するAPIのGETメソッド）

**from** integer
minimum: 1
maximum: 100

**to** integer
minimum: 1
maximum: 100

『from < to』という暗黙的な仕様（制約）を捉える

from, toのパラメータ単体で、
最小、最大の値の範囲を意識したテスト

**明示的な仕様から設計したMBTモデル**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | 有効/無効 | ☑ | ☑ | ☑ | ☑ | ☑ |
| | | | 1 | 2 | 3 | 4 | 5 |
| 条件 | from | 0 (最小値-1) | Y | N | N | N | N |
| | | 1 (最小値) | N | Y | N | N | N |
| | | 50 (中間値) | N | N | Y | N | N |
| | | 100 (最大値) | N | N | N | Y | N |
| | | 101(最大値+1) | N | N | N | N | Y |
| 動作 | Success | | - | X | X | X | - |
| | Error | | X | - | - | - | X |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | 有効/無効 | ☑ | ☑ | ☑ | ☑ | ☑ |
| | | | 1 | 2 | 3 | 4 | 5 |
| 条件 | to | 0 (最小値-1) | Y | N | N | N | N |
| | | 1 (最小値) | N | Y | N | N | N |
| | | 50 (中間値) | N | N | Y | N | N |
| | | 100 (最大値) | N | N | N | Y | N |
| | | 101(最大値+1) | N | N | N | N | Y |
| 動作 | Success | | - | X | X | X | - |
| | Error | | X | - | - | - | X |

from, toとパラメータを組み合わせたテストケースを生み出している

**暗黙的な仕様から設計した新たなMBTモデル**

| | | | | | |
|---|---|---|---|---|---|
| | | 有効/無効 | ☑ | ☑ | ☑ |
| | | | 1 | 2 | 3 |
| 条件 | from , to | from < to | Y | N | N |
| | | from = to | N | Y | N |
| | | from > to | N | N | Y |
| 動作 | Success | | X | - | - |
| | Error | | - | X | X |

**「パラメータ間の大小関係」を持つ場合のテストとして形式知化する**

# 知見を形式知化するアプローチ

VERISERVE

## ➢ テスト設計者が持つ知見

- 例えば、熟練したテスト設計者は暗黙的な仕様を捉えることが出来る
- 作成したテストケースをリバースモデリングすることでMBTで適用可能なように形式知化する



WebAPIテストでの事例を紹介

## ➢ 過去不具合から得た知見

- 例えば、**重大な不具合**の一部にはテスト観点として漏れていたものがある
- 不具合の発生条件をモデリングしMBTで適用可能なように形式知化する

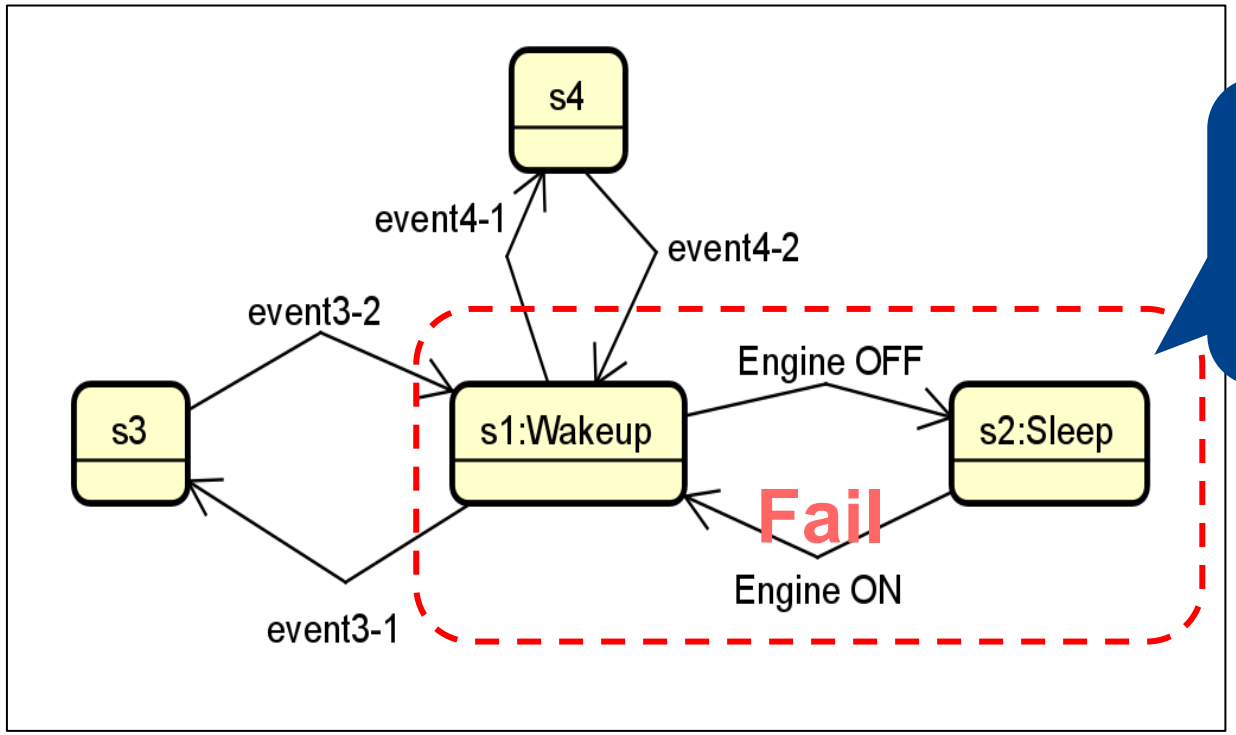
**状態遷移テストでの事例を紹介**

# 状態遷移における過去不具合

VERISERVE

- １つの状態遷移モデルから、各状態に遷移できることを確認するテストケースを生成したが、以下の不具合はそのテストケースからは検出できなかった

Summary
“Sleep” から“Wakeup”への状態遷移に失敗することがある




遷移時に
**不揮発性メモリ**
に読み書き処理をする

MBTモデル: 状態遷移モデル
（SUT: Automotive system)

* SUT : System under test

# 何故MBTで検出できなかったか？

VERISERVE

- 不揮発性メモリの信頼性の問題でデータが欠損した
- データ読み込み処理に失敗し遷移不可になった

不具合の原因

**s1とs2の遷移を複数回確認するテストケースが無かった**

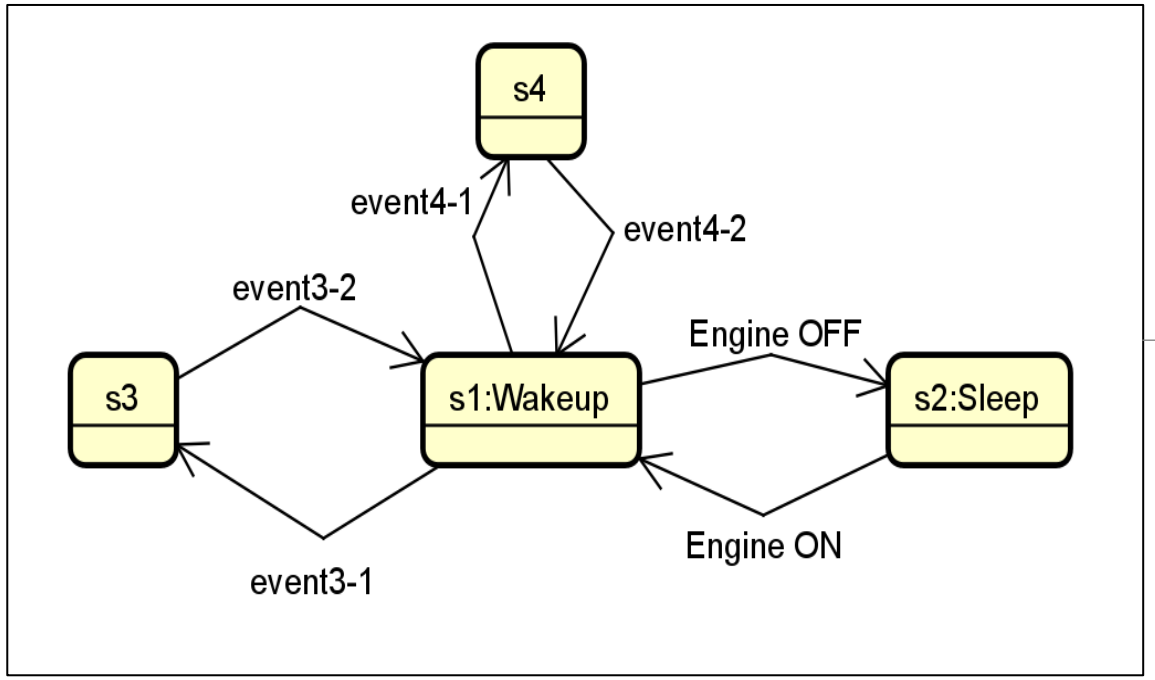
全ての遷移パスを網羅

| # | from | to |
|---|---|---|
| 1 | s1:Wakeup | s2:Sleep |
| 2 | s2:Sleep | s1:Wakeup |
| 3 | s1:Wakeup | s4 |
| … | … | … |

MBTモデルから作られたテストケース

# 複数のモデルを統合する必要がある

VERISERVE




状態遷移モデル

統合

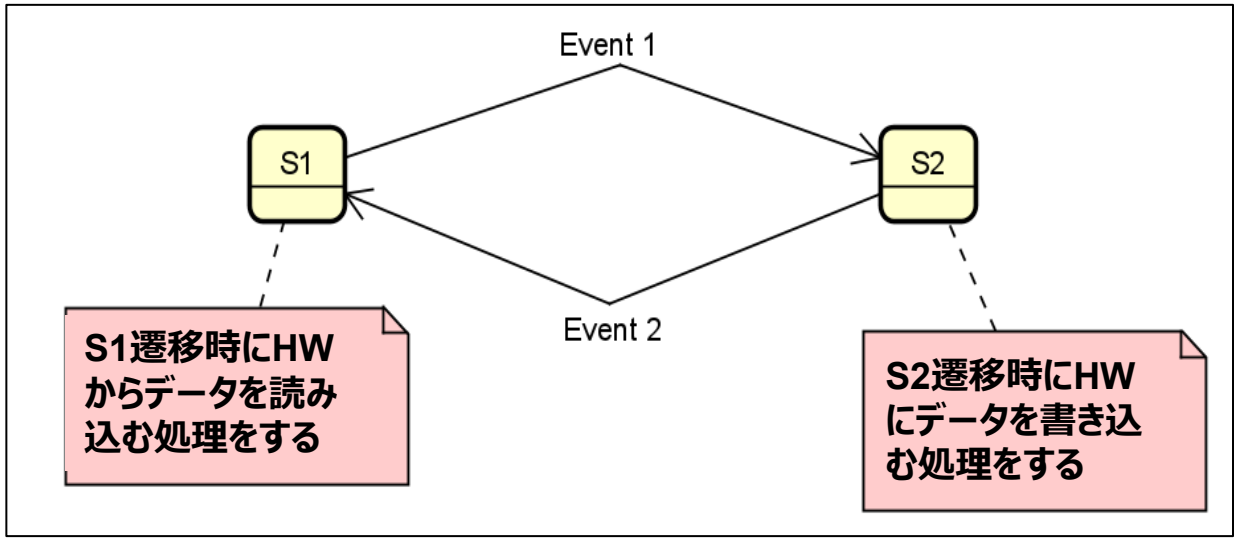
（不揮発性メモリに由来する）
**エラー確率モデル**

テストケース生成

"Sleep"と"Wakeup"の状態遷移を確率モデルを考慮して複数回試行する

# 過去不具合をベースにしたテストケース生成ロジック

VERISERVE




**もし、状態遷移モデルが上記の特性を持っているなら、以下のテストケースを生成する**

| テストステップ | 期待動作 | 試行回数 |
|---|---|---|
| S1 -> S2 -> S1 | S1に遷移し、処理が開始されること | N (>1) |

Nは、確率モデルのPの値に基づいて決定する

# 4. MBTツールの紹介

# ツール概要



## ➢ 目的

- MBTの社内普及

## ➢ コンセプト

- 手軽に手元でMBTを試すことができることで、MBTを使ってみようというモチベーションをアップさせる

## ➢ 特徴

- ブラウザ上でMBTモデリングからテストケース生成までできる
- 蓄積したテストパターン、テストケース生成ロジックがMBTツールに実装されている

# ということで、ベリサーブはMBTを本気でやります！
# テスト会社で全社エフェクトする技術開発に関わりたい方、ご一緒しませんか？

## We are hiring!!

品質を創造する

VERISERVE

本日はお時間をいただき、ありがとうございました。
貴社サービス・製品の品質向上に、
ベリサーブの全ノウハウを駆使してご支援をさせていただきたく考えております。
何卒よろしくお願いいたします。

**お問い合わせ**

株式会社ベリサーブ　ベリサーブ　検索
広報・マーケティング部
<vs.marketing@veriserve.co.jp>
TEL：050-3640-8194
東京都千代田区神田三崎町3-1-16 神保町北東急ビル 9F