

# ソフトウェア品質シンポジウム2022

## 暗黙的になりがちな開発仕様パターンを活用して モデルベーステストを改善する

発表資料

2022年9月8日（木）

○蛭田 恭章<sup>1</sup>, 須原 秀敏<sup>1</sup>, 藤田 真広<sup>1</sup>, 西 康晴<sup>2</sup>  
<sup>1</sup> 株式会社ベリサーブ, <sup>2</sup>電気通信大学 大学院情報理工学研究科

1. 本研究の背景と狙い
2. 実施概要
3. パターン蓄積フェーズ ※パターン = 暗黙的になりがちな仕様に共通するパターン
4. パターン活用フェーズ
5. 考察
6. まとめ

# 1. 本研究の背景と狙い

---

- ソフトウェアテストや品質保証に対して、ソフトウェアのリリースのスピードを保ちつつも、品質を向上するための貢献（自動化など）が求められている
  - World Quality Report 2021-22(Capgemini)では、2020年から以下の調査項目が追加されている
    - ◆ ソフトウェアリリースのスピードを保ちつつも品質を向上させる
    - ◆ 高品質を達成するためにチーム全員をサポートする
    - ◆ QAやテストプロセスを自動化でよりスマートにする
- 上記要求に対応する技術として、MBTに注目した
  - MBTはテスト設計仕様として表したMBTモデルから直接テストケースを自動生成する技術であり、テスト実行まで自動化されることも多い
  - またアジャイル開発などでは、ソフトウェアの変更に伴う動作保証を目的としたテストが、高い頻度で自動で実施される
  - MBTによる自動化のテストを拡大することは、頻繁に実行されるテストの動作保証の範囲を広げられると考える

UML/SysMLなどのモデル表現の開発仕様をMBTの入力することも多い



開発仕様

テスト設計



MBTモデル

自動生成

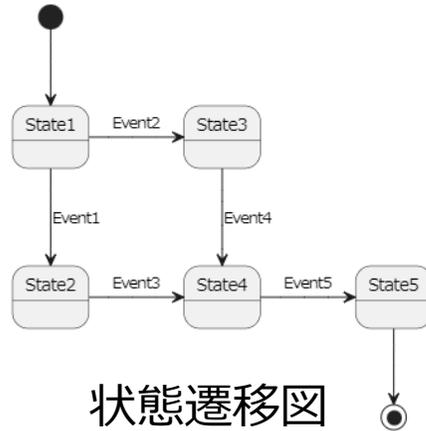


テストケース

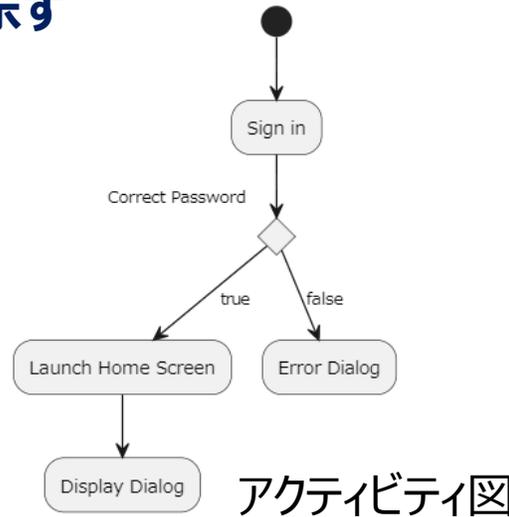
テスト自動実行



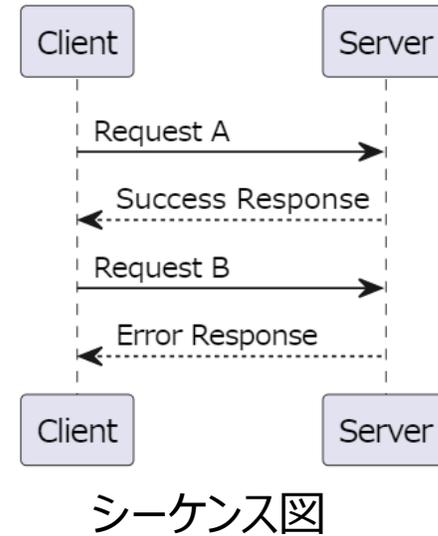
- MBTモデルには、さまざまな表現方法があるが、どんな側面をテストするかによって使用するモデルも異なる
- 一般的に使用されるMBTモデルを示す



状態遷移図



アクティビティ図



シーケンス図

		有効/無効				
		1	2	3	4	
条件	条件1	A	Y	Y	N	N
		B	N	N	Y	Y
	条件2	X	Y	N	Y	N
		Y	N	Y	N	Y
動作	動作1	X	-	-	X	-
	動作2	-	X	X	-	-

デシジョンテーブル

- MBTの性質上、テストすべきことは明示的にMBTモデルに盛り込む必要がある
- アジャイル開発では重要なところは仕様モデルを描くこともあるが、どうしても暗黙的な仕様は出てくる
- 暗黙的な仕様をMBTモデルに盛り込むことで、自動テストでの動作保証の範囲が広がり、開発スピードも保ちつつも、品質を向上することへの貢献を期待できる
- しかし、暗黙的な仕様を捉えるには経験やスキルが必要になるため容易ではない



本発表で対象にするMBTの課題

MBTモデル開発時、暗黙的な仕様をどう把握するか

## ➤ WebAPIのticket-statusというパラメータに関する仕様

```
parameters:  
  ticket-status:  
    name: ticket-status  
    in: query  
    type: string  
    enum:  
      - open  
      - in_progress  
      - done
```

ticket-statusとパラメータに対して、openとin\_progressとdoneという値を指定できることが分かる

この仕様から、仕様内の各値と仕様外の値を指定する条件を盛り込んだMBTモデルを開発する

		有効/無効	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
			1	2	3	4
条件	ticket-status	open	Y	N	N	N
		in_progress	N	Y	N	N
		done	N	N	Y	N
		start	N	N	N	Y
動作	success	X	X	X	-	
	fail	-	-	-	X	

ticket-statusは複数の値を指定できる仕様も存在した！

暗黙的な仕様

左記の仕様があるのであれば、ticket-statusに以下のように複数の値を指定した場合の条件をMBTモデルに盛り込む必要がある

- in\_progress, done
- open, in\_progress, done

- 仕様に書かれていないけど、テストケースには書かれていることがある
- テストエンジニアは暗黙的になりがちな仕様の共通的なパターンを知っているのではないかと考えた
- そして、そのようなパターンを蓄積し、活用できれば暗黙的な仕様を盛り込んだMBTモデルを開発でき、結果的にテストケースを増やせて、動作保証範囲を拡大できるのではないかと考えた
  
- **今回の取り組み**
  - パターン蓄積フェーズ
    - ◆ 開発仕様にて暗黙的になりがちな仕様に共通するパターンを蓄積する
  - パターン活用フェーズ
    - ◆ 暗黙的になりがちな仕様に共通するパターンを活用して、新たなMBTモデル開発、テストケース生成ができるかを試す

## 2. 実施概要

---

## ➤ パターン蓄積フェーズ

- WebAPI仕様を対象に、過去に作成された4プロジェクトのテストケースを分析する
- 仕様書にはなくテストケースにはあった仕様を、暗黙的な仕様と捉えて抽出する
- その中で共通性のあるものをパターンとして蓄積する

## ➤ パターン活用フェーズ

- パターン蓄積フェーズとは別の2プロジェクトにてパターンを活用したテスト設計を実施する
- パターンを使うことで新たなMBTモデルの開発、新たなテストケース生成ができるかを確認する

## ➤ 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)

GET /users/{owner\_name}/repositories/{repository\_name}/test\_cases Get test models and test case specifications.

Get test models and test case specifications. If the {owner\_name} is `sample_user` and the {repository\_name} is `MyRepository`, specify the following URL for API call: `https://gihoz-api.veriserve.co.jp/api/v1/users/sample_user/repositories/MyRepository/test_cases`.

Parameters

Name	Description
owner_name * required string (path)	User name of the repository owner
repository_name * required string (path)	Repository name

Responses

Code	Description	Links
200	List of test cases for {repository_name}.	No links

Media type: application/json

Example Value | Schema

```
{
  "id": 10,
  "type": "test_case",
  "attributes": {
    "name": "テストケースサンプル",
    "user_id": 10,
    "test_type": "state_transition",
    "uuid": "80ca3ab9-75ca-41c8-a5ff-26767755a5e0",
    "update_at": "2021/06/28 14:17"
  }
}
```

(swagger.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_name} is `MyRepository`, specify the following URL for API call:
      `https://gihoz-api.veriserve.co.jp/api/v1/users/sample_user/repositories/MyRepository/test_cases`."
    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"
```

### 3. パターン蓄積フェーズ

---

## 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

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

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

暗黙的になりがちな仕様に  
共通するパターン

パラメータ間の大小関係

共通性

2つのパラメータの値において大小関係を持つ必要があるという制約を持つ

暗黙的な仕様

maxPriceの値は  
minPriceの値より  
大きい値で指定する

endの値は  
startの値より  
大きい値で指定する

maxCountの値は  
limitの値より  
大きい値で指定する

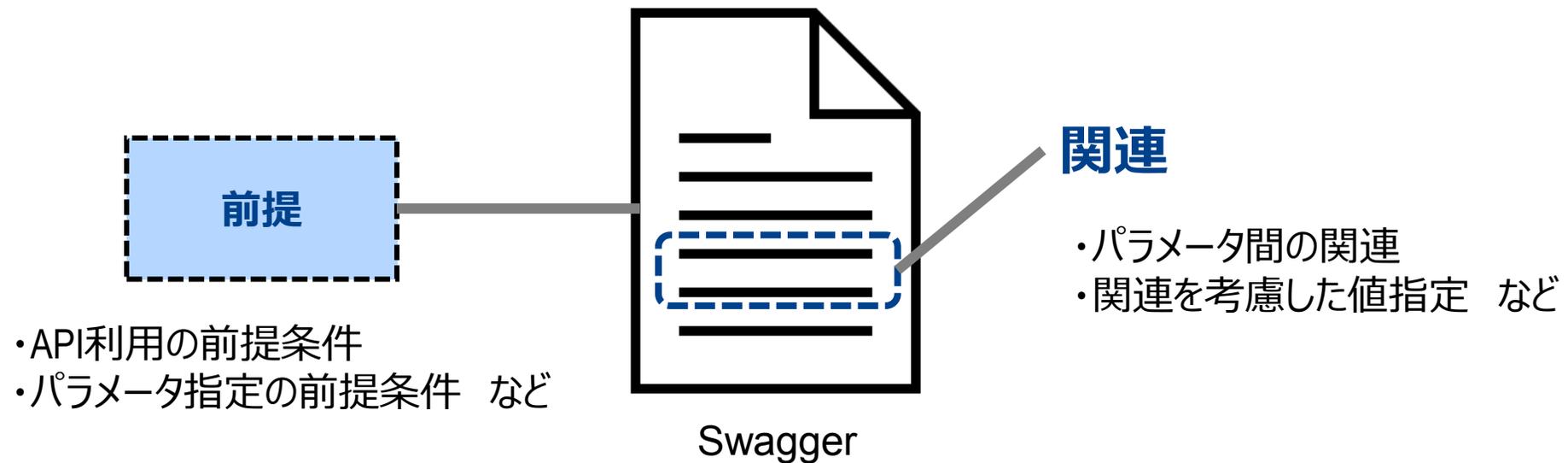
Swaggerに  
記載された仕様

パラメータ  
- maxPrice  
- minPrice  
- ...

パラメータ  
- start  
- end  
- ...

パラメータ  
- maxCount  
- limit  
- ...

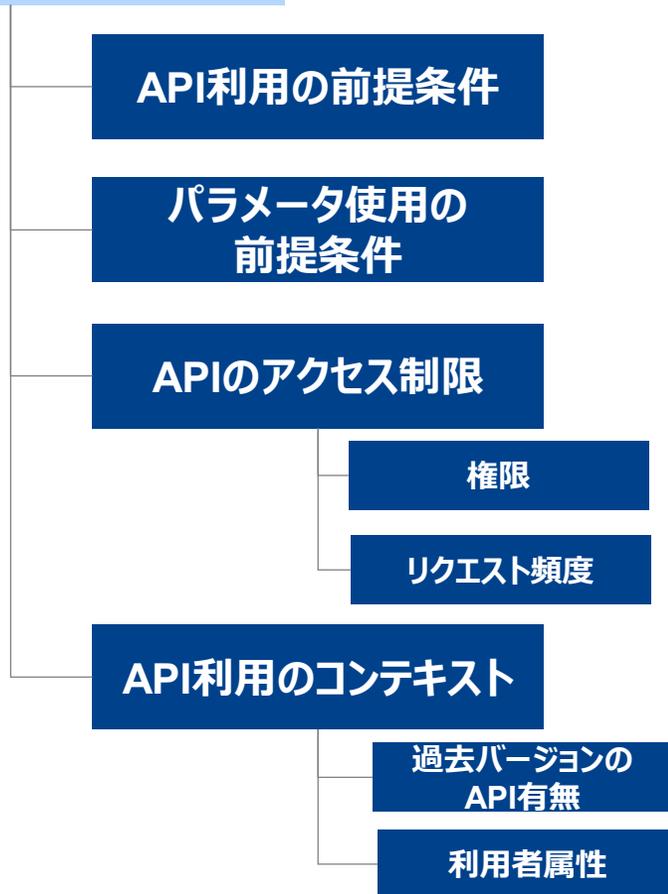
- **Swaggerに定義された仕様の「前提」に関わるパターン**
  - Swaggerで定義する仕様のスコープ外に当たるため暗黙的な仕様になる可能性がある
- **Swaggerに定義された仕様間の「関連」のパターン**
  - Swaggerで形式的に表現しきれないため、暗黙的な仕様になる可能性がある



➤ 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

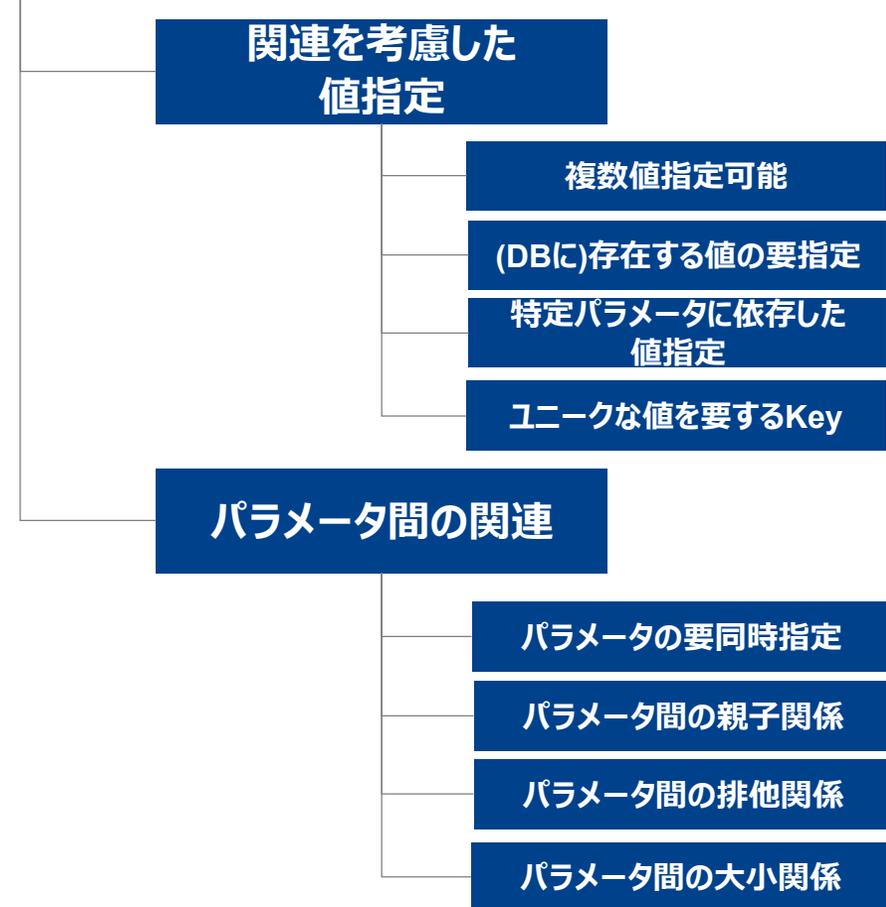
6パターン



## 『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



## 4. パターン活用フェーズ

---

## ➤ 従来法、パターン活用法でそれぞれテスト設計(MBTモデル開発～テストケース生成)を以下のAPIで実施した

### ■ テスト設計方法

#### ◆ 従来法

- Swaggerの仕様のみを入力としテスト設計を実施

#### ◆ パターン活用法

- パターンを活用することで把握できた暗黙的な仕様も踏まえてテスト設計を実施

### ■ テスト設計対象のAPI

#### ◆ イベントを更新するAPIのPUTメソッド

- 指定するパラメータ数:4, Request Bodyで使用するKeyの数:25

#### ◆ 指定した範囲のスコアの値を取得するAPIのGETメソッド

- 指定するパラメータ数:21, Request Bodyで使用するKeyの数:0



### テスト設計者

テスト設計経験：約2年  
APIテスト経験：約1年  
対象ドメイン経験：約1年

## ➤ テスト設計者は以下の内容でパターンを活用した

- 各パターン(17パターン)を基にして仕様を調査した
  - ◆ パラメータが複数ない、新規開発のAPIなどの場合は調査不要なパターンも存在した
- 調査方法として、以下の順で進める
  1. Swagger内に形式的ではないが、コメントとして仕様が記載されていないかを確認
  2. Swagger以外の文書（開発者との打ち合わせ議事録など含む）に仕様が記載されていないかを確認
  3. 開発者に問い合わせ確認
- 入手できた情報（仕様）を基にMBTモデルを開発し、テストケースを生成する

- 従来法とパターン活用法で動作保証範囲を比較した
  - 動作保証範囲はMBTモデル数とテストケース数で比較した

	MBTモデル数（個）				テストケース数（件）			
	従来法	パターン活用法	新たに開発したMBTモデル数	既存MBTモデルの変更数	従来法	パターン活用法	新たに開発したMBTモデルから生成されたテストケース数	変更MBTモデルから新たに生成されたテストケース数
イベントを更新するAPIのPUTメソッド	19	<b>21</b> (約11%増)	2	1	76	<b>84</b> (約11%増)	7	1
指定した範囲のスコアの値を取得するAPIのGETメソッド	17	<b>20</b> (約18%増)	3	1	67	<b>77</b> (約15%増)	8	2

以上の通り、パターン活用法は従来法と比較して、  
MBTモデル数、テストケース数が約11～18%増加する結果となった

## 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のパラメータ単体で、  
最小、最大の値の範囲を意識したMBTモデル

## 従来法で作られた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
		1 (最小値)	N	Y	N	N
		50 (中間値)				
		100 (最大値)				
		101 (最大値+1)				
動作	Success					
	Error					

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



「パラメータ間の大小関係」  
というパターンを活用し、  
暗黙的な仕様を調査する



『from < to』という暗黙的な  
仕様（制約）を捉える

パターン活用により  
from, toとパラメータを組合せたMBT  
モデルを新たに開発する



## パターン活用法で新たに作られたMBTモデル

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

## 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



## 従来法で作られたMBTモデル

			有効/無効			
			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
			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



## パターン活用法により変更を加えたMBTモデル

			有効/無効				
			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
			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

## 5. 考察

---

## ➤ パターン活用フェーズで見つかった暗黙的な仕様について

※パターン = 暗黙的になりがちな仕様に共通するパターン

### ■ 仕様記述について

- ◆ 今回把握できた暗黙的な仕様のうち、Swagger内にコメントで記載されていた仕様も存在した
  - 例で紹介したevent\_idに対する「ユニークな値を要するKey」が該当する
  - Swaggerで形式的記述するには制限があるので、コメントなどで制約等の仕様を補っていると考えられる
- ◆ 「パラメータ間の大小関係」については文書に記述されておらず、開発者に問い合わせることで把握した
  - 例で紹介したfrom/to以外にも、min/maxやstart/endなどが該当した
  - 直感的に大小関係を認識できるパラメータだったので、記述を省くことが多いと予想される

### ■ 不明な仕様

- ◆ 「特定パラメータに依存した値指定」というパターンをきっかけに把握した暗黙的な仕様について開発者に質問した
- ◆ 回答結果は、明確な仕様が分からず、「動作としてはおそらくエラーになるはず」という内容だった
- ◆ その後、テスト実施した結果の振る舞いを開発者に伝えるということがあった
- ◆ 仕様が明確でない場合、テスト実施した結果から仕様を見直すことも考えられる
- ◆ その場合でもMBTにより仕様変更の対応が容易になることも期待できる

※パターン = 暗黙的になりがちな仕様に共通するパターン

## ➤ テスト設計者へのヒアリングを通して分かったこと

### ■ 時間的な側面

- ◆ パターン活用法でも、従来法と比較をして大幅に時間がかかるようなことはなかった
- ◆ これまでも経験に基づきに暗黙的な仕様を調査していたため、全体的な作業時間として増える事はなかったと思われる
- ◆ パターンにより何を意識して調査するかが明確になった点良かったというコメントをもらった

### ■ パターン活用の難易度

- ◆ 作業開始時にパターンの解釈について質問をされることがあり、その場で追加説明を行った
- ◆ パターン活用により明らかにできた仕様もあったが、見逃している仕様もあるかもしれないというコメントもあった
- ◆ 今回正しくパターンを活用できていたかの側面では検証できていないため、その点は今後の課題である

※パターン = 暗黙的になりがちな仕様に共通するパターン

## ➤ パターン活用の有効性の更なる検証

- 今回の取り組みでは従来法とパターン活用法を比較してMBTモデル、テストケースが増加したかの観点でしか検証していない
- 以下の点におけるパターン活用の有効性の検証は今後の課題である
  - ◆ 経験に基づいて暗黙的な仕様を把握する場合と比較してどう有効なのか
  - ◆ またテスト設計者間のバラつきを抑制することによるどの程度有効なのか

## ➤ 対象とするシステムや開発仕様モデルの拡大

- 今回の取り組みでは、1つのシステムにおけるWebAPIテストに限定した実験であった
- 異なるシステムにおけるWebAPI仕様においては他のパターンも存在すると予想される
- また状態遷移モデルやシーケンスモデルにおいても、暗黙的になりがちな仕様も存在すると考えられる
- 上記を対象としたパターンの蓄積が今後の課題である

## 6. まとめ

---

## ➤ 今回の取り組みと成果

### ■ パターン蓄積フェーズ

- ◆ 取り組み：既存のテストケースを分析し、WebAPI開発仕様における暗黙的になりがちな仕様に共通するパターンを蓄積した
- ◆ 成果：WebAPI開発仕様（Swagger）にて、暗黙的になりがちな仕様に共通する17パターンを蓄積できた

### ■ パターン活用フェーズ

- ◆ 取り組み：パターン蓄積とは別のプロジェクトにてパターンを活用したテスト設計を実施した
- ◆ 成果：従来法と比較してパターン活用法では、暗黙的な仕様をMBTモデル/テストケースに盛り込むことができた
  - MBTモデル数、テストケース件数を約11～18%増やすことができた

## ➤ 今後について

- パターンの有効性についての検証を継続する
- 異なるシステムのWebAPIや状態遷移モデルやシーケンスモデルを対象としたパターンの蓄積と活用を継続する

品質を創造する

VERISERVE

ご清聴ありがとうございました