ちょ・す cho-su [動サ五]
意味：触れる。さわる。操作する。

参考情報

ちょしてみよう！ テストスイートモデリング

～技法と観点を活用しやすくするカタマリー～

下浅 大輔（テクノラボ）
情野 吉紀（JaSST Hokkaido 実行委員会）
水野 昇幸（JaSST Hokkaido 実行委員会）

Powered by IBURI

参考情報メニュー

- テスト開発におけるモデリング
- テストケースの粒度
- テストカタマリーとは
- テストカタマリー表記法
- キガカリー＆ブンルイー
- テスト方針
- カタマリー例
- 参考リンク

テスト開発におけるモデリング

モデリングとは

ある事象の抽象化されたコンセプトモデル・グラフィカルモデル・あるいは数理モデルを作るプロセスである。

科学の様々な分野において、各々特化した科学的モデリングのための科学的方法、技術、理論が蓄積されている。

科学的モデリングによって事象を構成する様々な要素が単純化され、容易に読み取れるようになる。

（ウィキペディアより）

モデリング言語

ADL

システムアーキテクチャの記述や表現のためのアーキテクチャ記述言語

UML

ソフトウエア・システムのための統一モデリング言語

他

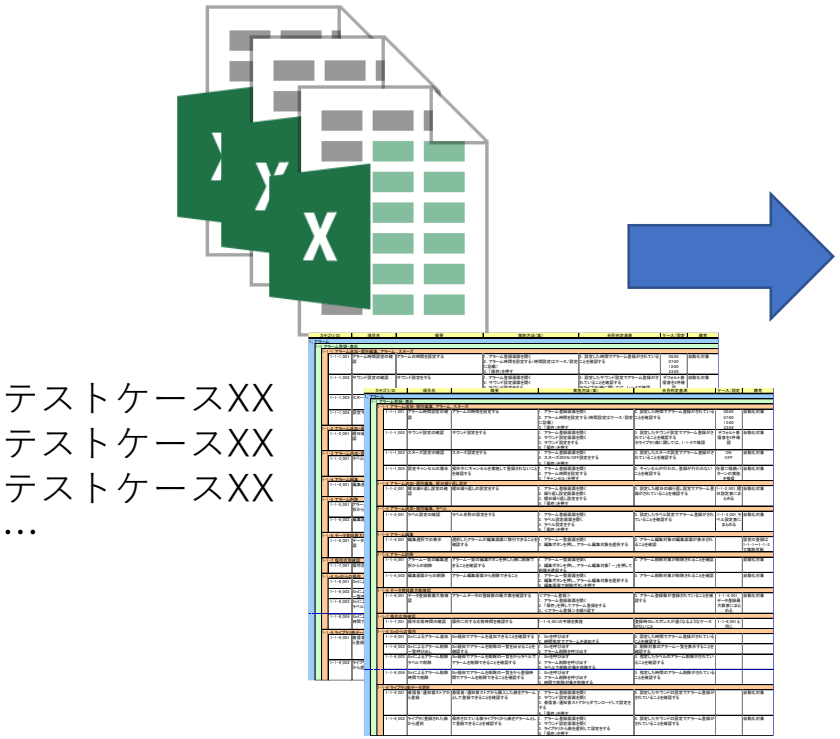
テストでも使われているモデリング

- クラシフィケーションツリー
- フローチャート
- 状態遷移
- 数式化

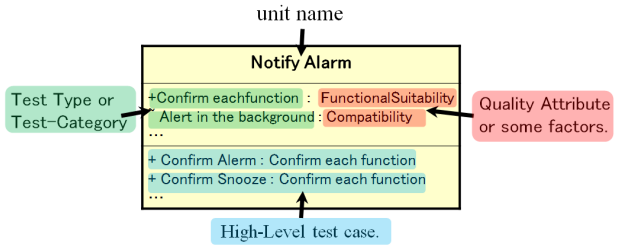
他

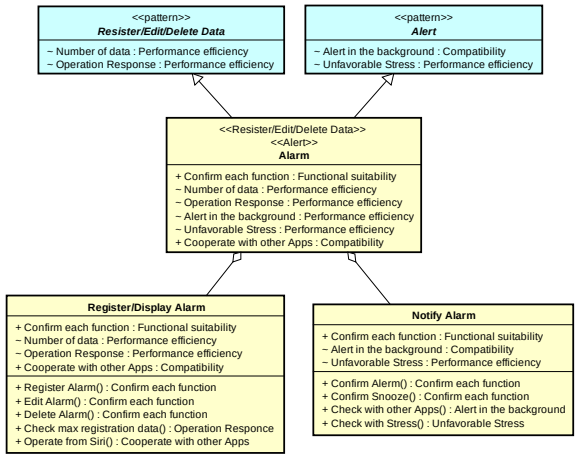
テストスイートモデル

テストケースが持つ情報構造をモデルで表したものを
「**テストスイートモデル**」と本ワークショップ内では呼びます。
今回のワークショップ内では、クラス図に似た表現でモデルを描きます。



テストケースXX
テストケースXX
テストケースXX
…





テストスイートモデル

「**テストスイートモデル**」はテストケースが持つ情報の構造を
モデルで表現したものです。※独自の用語です。
「テストスイートアーキテクチャ」という、テストシステムアーキテクチャ」
と対となる概念として、以下の論文にて紹介されております。

＜テストスイートアーキテクチャについて記載している論文＞
Y.Nishi, “Combinatorial Test Architecture Design Using Viewpoint diagram”,
Workshop on Metrics and Standards for Software Testing (MaSST)

Y.Nishi, “Design principles in Test Suite Architecture,”
International Workshop on Software Test Architecture (InSTA 2015)

上記テストスイートアーキテクチャを参考に、テストスイートモデルという
独自の概念・名前を使用しております。

テストにおけるモデリングアプローチ

UTP

UML Testing Profile
※Unit Test Patternではありません。

NGT/VSTeP

NGT: Notation for Generic Testing
VSTeP: Viewpoint-based Software Test Engineering Process
※電気通信大学にしさんが提案。テスト設計手法がVSTePで、その手法内で使用している記法がNGTとなります。

テストケースの粒度

レベルテストケース

高位レベルテストケース（high level test case）

具体的な（実行レベルの）入力値や予測結果を使わないテストケース。論理演算子は使用するが、値のインスタンスは未定義や使用不可であるといった状態にある。

論理的テストケース（logical test case）

high level test caseを参照のこと。

抽象的テストケース（abstract test case）

high level test caseを参照のこと。

低位レベルテストケース（low level test case）

入力データと期待結果が具体的（実装レベル）なテストケース。高位レベルのテストケースからの論理演算子は、論理演算子に相当する実際の値に置き換えられる。

具体的テストケース（concrete test case）

low level test caseを参照のこと。

（「JSTQB 用語集」より）

具体的テストケースと論理的テストケース

具体的テストケース

テスト担当者がテストケース（すべてのデータ要件を含む）を実行し結果を検証するために必要な特定の情報と手順のすべてを提供する。

論理的テストケース（ロジカルテストケース）

テスト対象とすべきものに関するガイドラインを提供し、テスト実行時に実データや手続きさえも変更することを可能にする。

（「JSTQB TAシラバス」より）

テストケースの粒度：仮に４つの段階を紹介

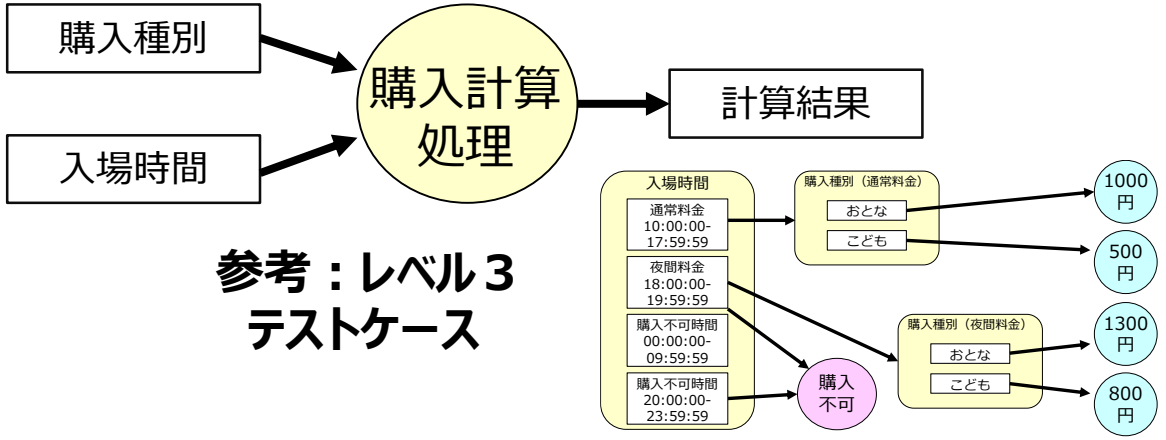
比較的抽象的なテストケース粒度では…？

レベル１：処理だけを特定したテストケース

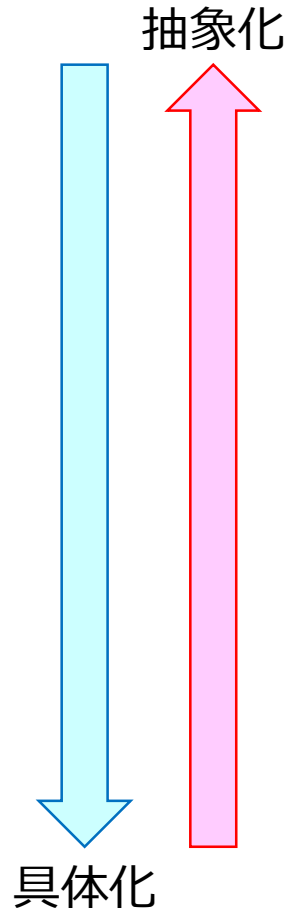
（例）購入計算処理のロジック/入出力を確認する

レベル２：入出力のパラメータ項目まで表現したテストケース

（例）購入種別と入場時間から計算結果を確認する



参考：レベル３テストケース

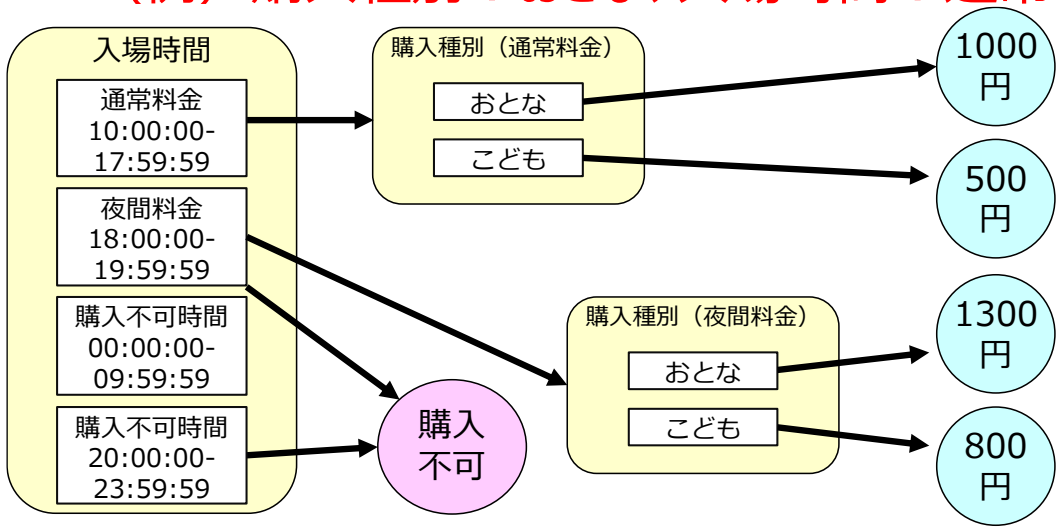


テストケースの粒度：仮に４つの段階を紹介

より詳細なテストケースでは…

レベル３：同値クラスを用いて組合せ論理条件（ルール）まで表現したテストケース

（例）購入種別：おとな、入場時間：通常料金時間　で結果1000円



（例）
CFDとDTで
関係性を図示

入力条件	#1	#2	#3	#4	#5	#6
1 購入種別						
1 おとな	－	－	○	○	－	－
2 こども	－	－	－	－	○	○
2 入場時間						
1 購入不可時間 00:00:00−09:59:59	○	－	－	－	－	－
2 通常料金時間 10:00:00−17:59:59	－	－	○	－	○	－
3 夜間料金時間 18:00:00−19:59:59	－	－	－	○	－	○
4 購入不可時間 20:00:00−23:59:59	－	○	－	－	－	－
期待結果	**#1**	**#2**	**#3**	**#4**	**#5**	**#6**
1 購入不可	○	○	－	－	－	－
2 1300円	－	－	－	○	－	－
3 1000円	－	－	○	－	－	－
4 800円	－	－	－	－	－	○
5 500円	－	－	－	－	○	－

レベル４：実際の値まで具体化したテストケース

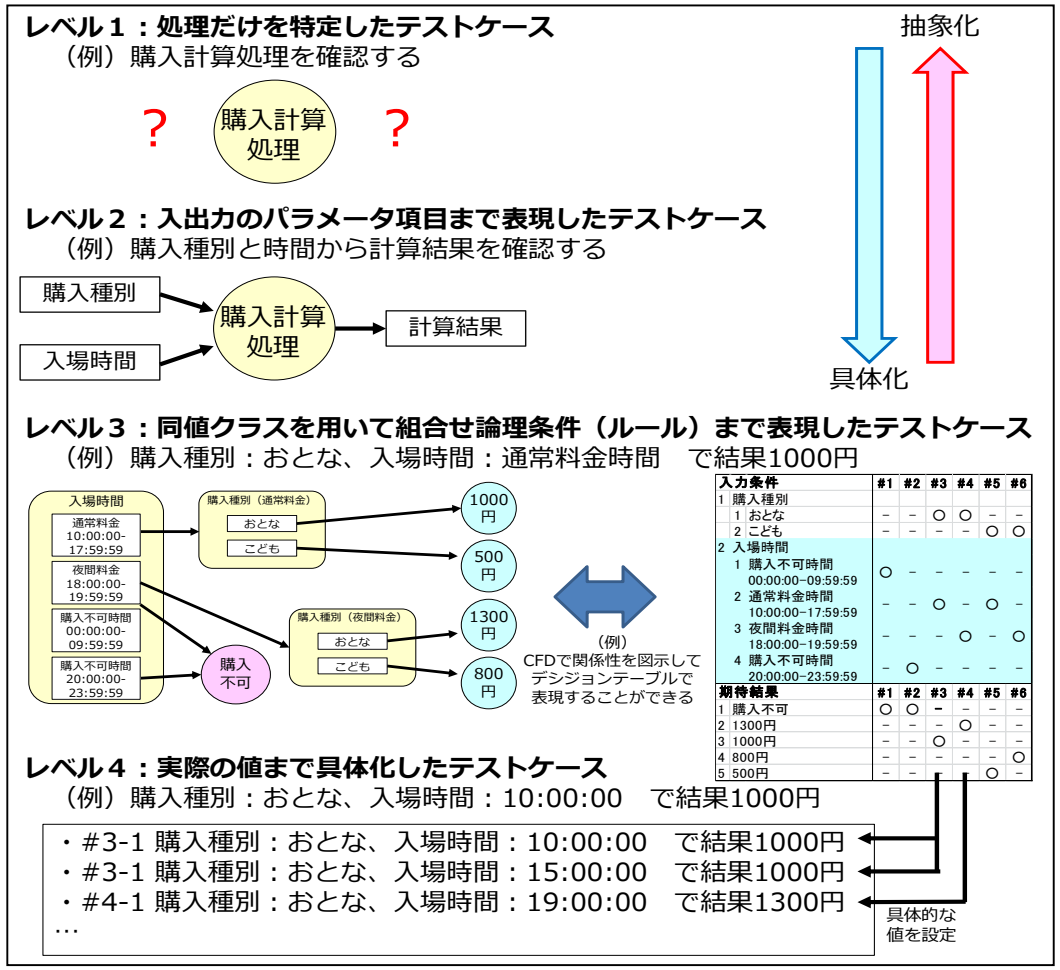
（例）購入種別：おとな、入場時間：10:00:00　で結果1000円

・#3-1 購入種別：おとな、入場時間：10:00:00　で結果1000円
・#3-1 購入種別：おとな、入場時間：15:00:00　で結果1000円
・#4-1 購入種別：おとな、入場時間：19:00:00　で結果1300円
…

具体的な
値を設定

テストケースの粒度：仮に４つの段階を紹介

全体を示すとこの通りになります。



各レベルの例

レベル１：購入計算処理のロジック/入出力を確認する

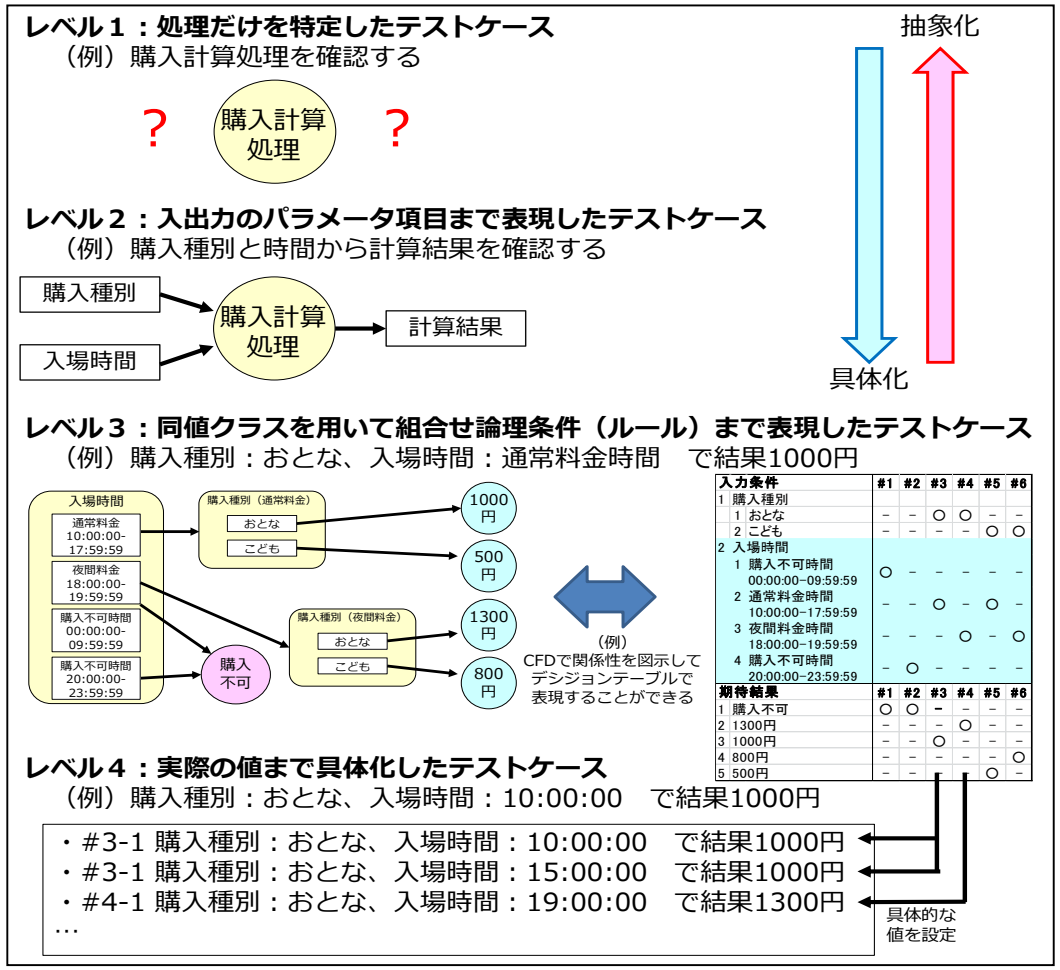
レベル２：購入種別と時間から計算結果を確認する

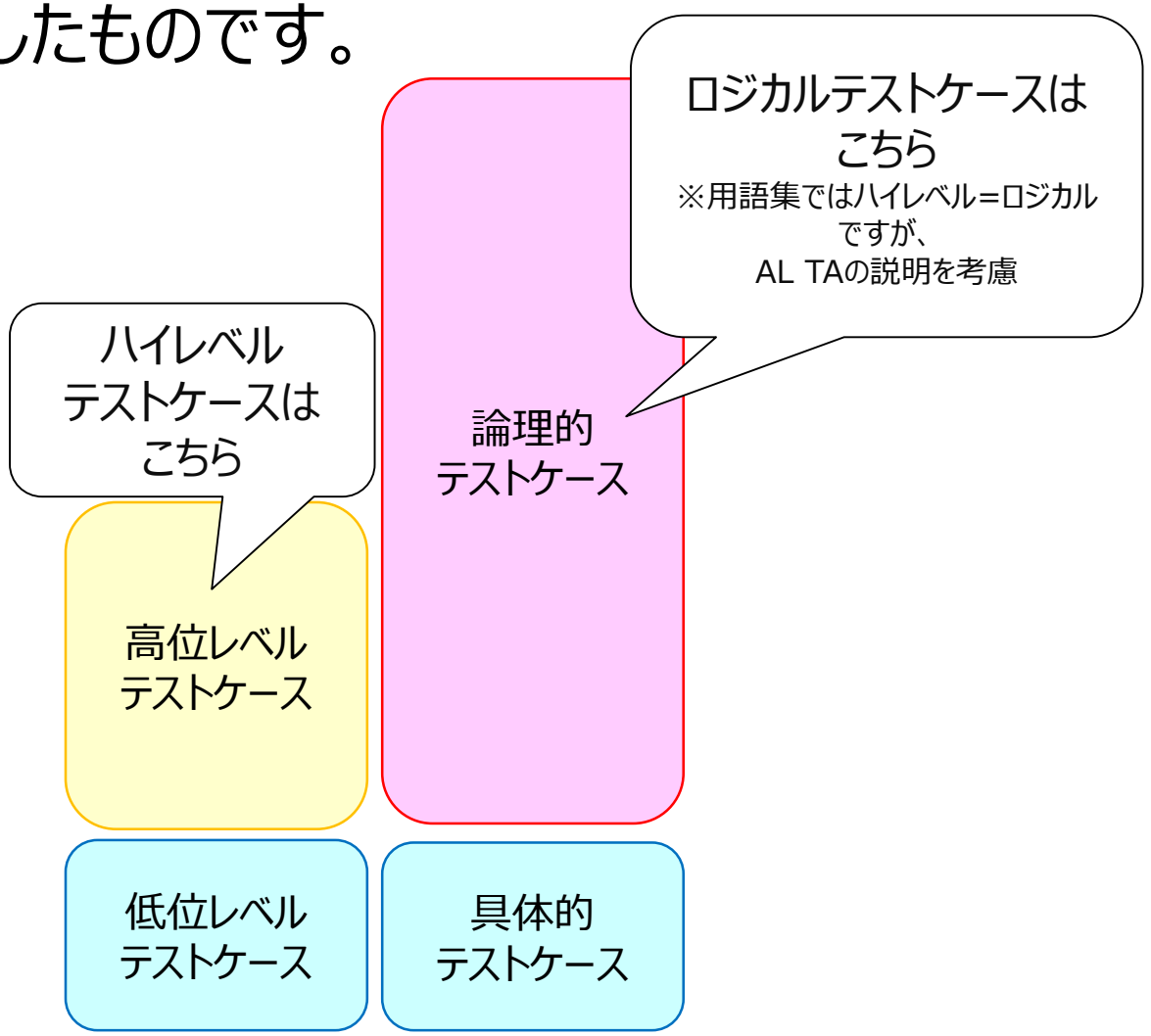
レベル３：購入種別：おとな、入場時間：通常料金時間で結果1000円

レベル４：購入種別：おとな、入場時間：10:00:00で結果1000円

テストケースの粒度と呼び名@ISTQB

ISTQB/JSTQBでの呼び名を整理したものです。





テストカタマリーとは

※注：本資料に出てくる以下の用語は独自に定義したものです

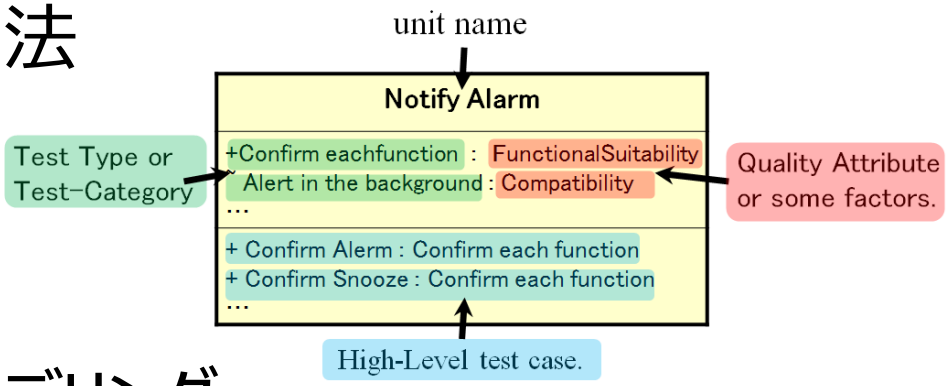
テストカタマリー/カタマリー

キガカリー、ブンルイー

テストカタマリーの概要

テストカタマリーとは

- STUDIO IBURIという謎の団体が打ち上げた記法
- テストカタマリー：テストケースの集合体（塊）を表現したもの
- テスト観点とテストカテゴリ、抽象的なテストケースを
  クラス図を利用して表現する記法

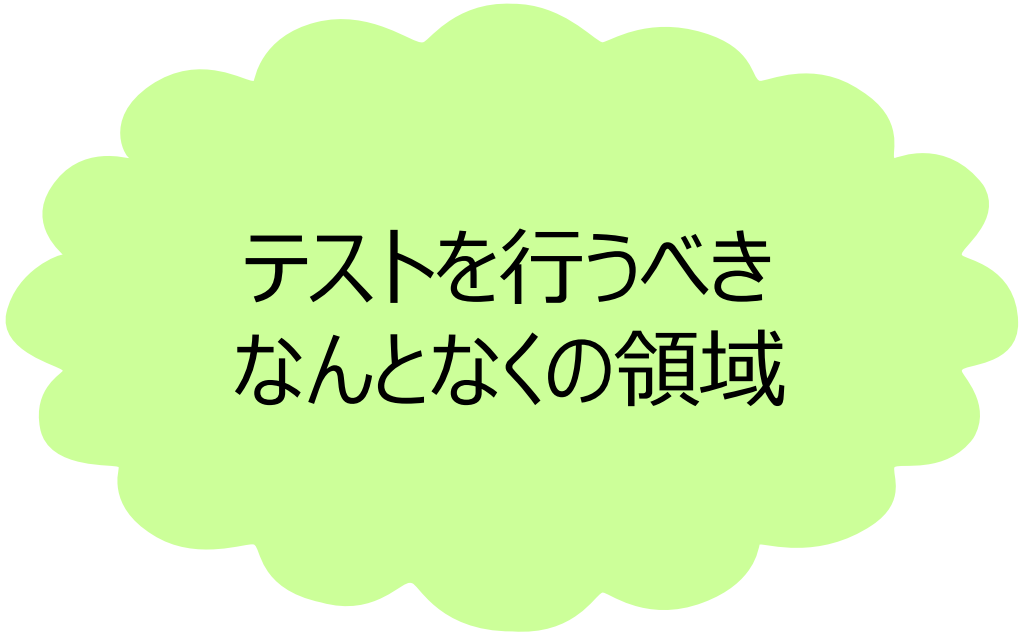


特徴

- UTPとVSTePの中間くらいのモデリング
- テストケースの概要の整理がメイン
- UMLモデリングツールが使えればそれでお手軽に描ける

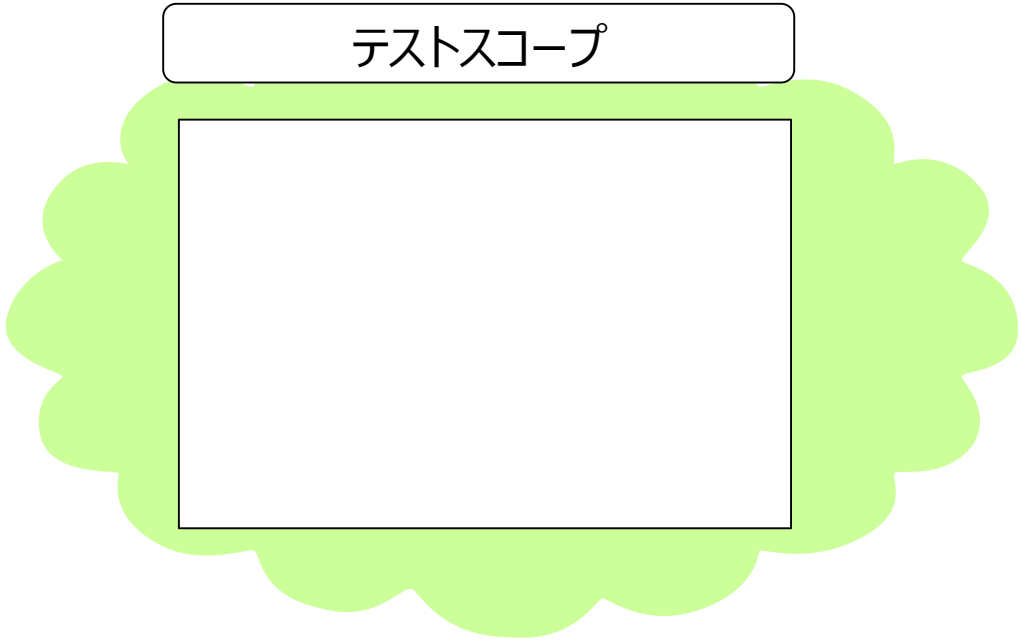
テストカタマリーのアイデア

テストカタマリーのアイデアの原型



テストカタマリーのアイデア

テストカタマリーのアイデアの原型

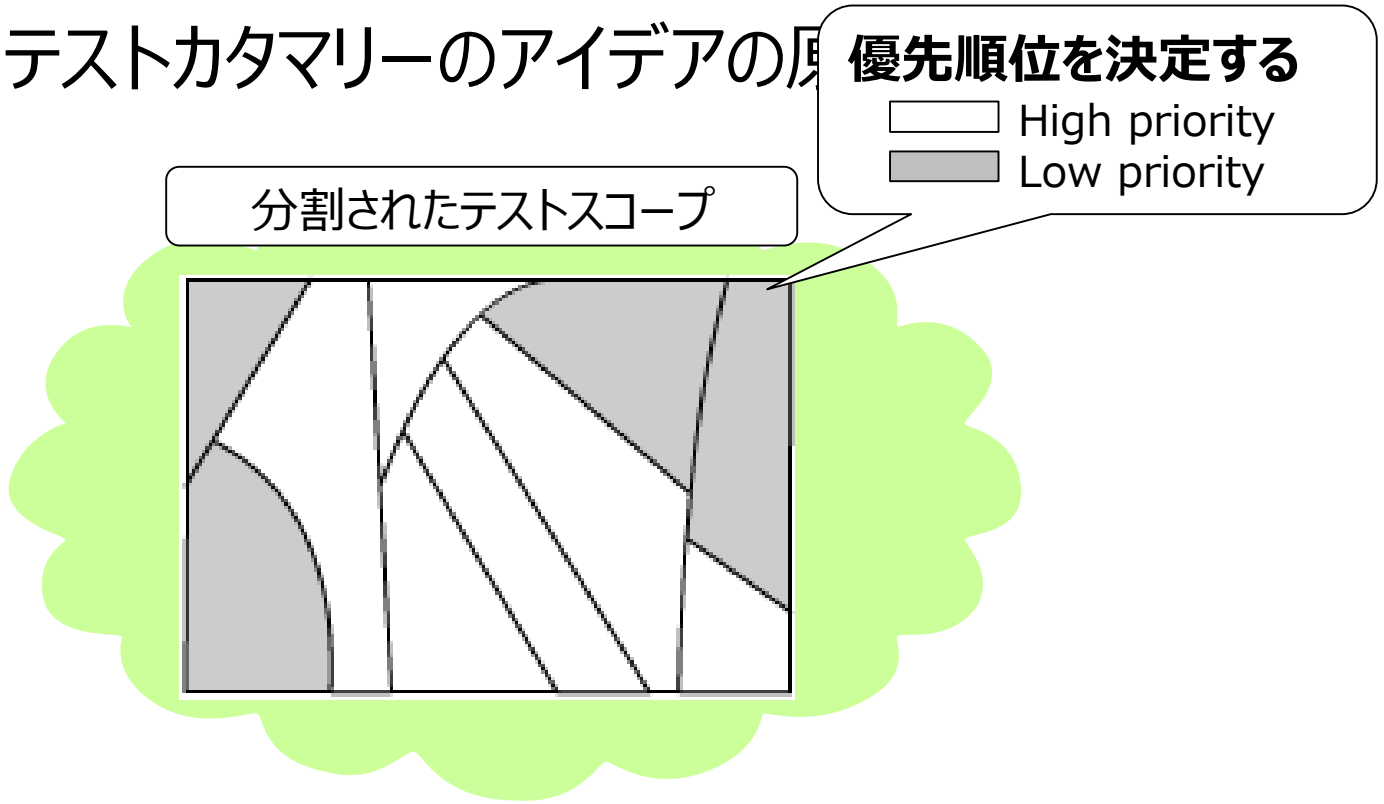


テストカタマリーのアイデア

テストカタマリーのアイデアの原型

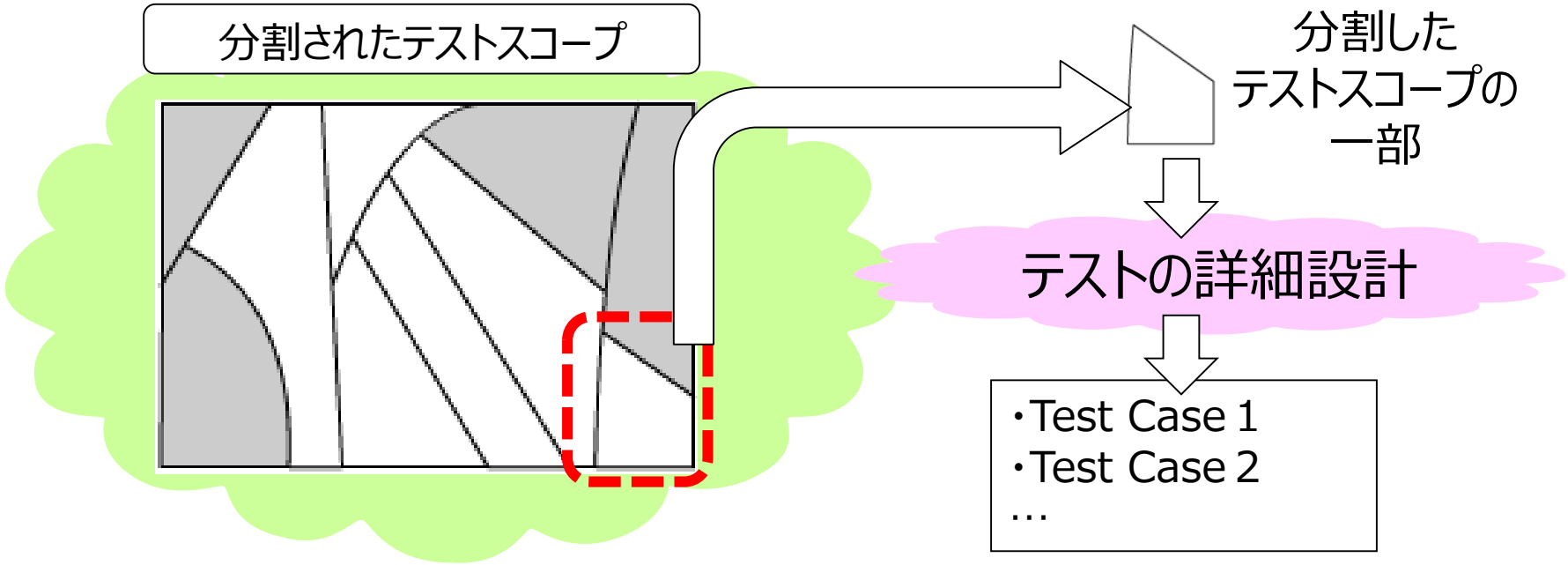
テストカタマリーのアイデア

テストカタマリーのアイデアの原



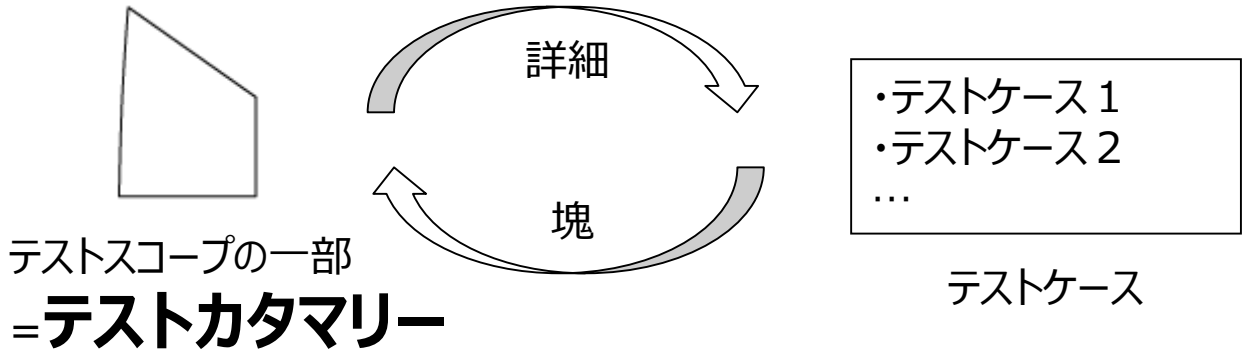
テストカタマリーのアイデア

テストカタマリーのアイデアの原型



テストカタマリーの概要

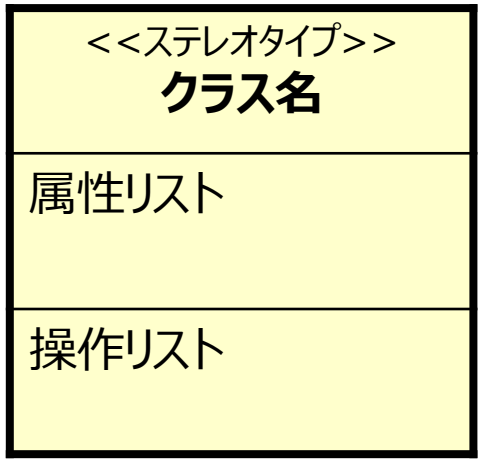
一連のテストケースを持つ
分割されたテストスコープを
「テストカタマリー」
と定義。



テストカタマリー表記法

テストカタマリー表記法：エンティティ表現

クラス図の記法を利用したテストカタマリー図の記法について概要を紹介する。



フィールド	クラス図	カタマリー
クラス名	クラス名を記述する。 継承もとの親クラスの情報を追加で表記することもある。 ステレオタイプを記述することで、クラスを種類ごとに分類したり、目印をつける。	カタマリー名を記述する。 ステレオタイプを使うことでカタマリーをパターンで分類することが出来る。
属性リスト	クラスが持っている情報群を表している。 型の指定と可視性を設定できる。 可視性　名前：型 = 初期値 { 制約条件 }	カタマリーが確認したい品質特性や気がかり、テストケースの目的等（キガカリーと呼ぶ）を記載する。 可視性 具体的なキガカリー：キガカリーの分類
操作リスト	クラスができることが書かれている。 型の指定と可視性を設定できる。 可視性 名前 (引数の名前：引数の型)：戻り値の型	カタマリーが持つ抽象的なテストケースがを記載する。 対応したいロジカルテストケース（抽象的なテストケース）に対し、確認したいキガカリーを割り当てる。 網羅基準ともなるテスト方針をあわせて定義可能。 可視性 ロジカルテストケース (テスト方針)：キガカリー

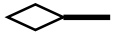
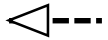
テストカタマリー表記法：可視性の表記

クラス図のとの違い

記号	可視性の種別	クラス図	カタマリー
+	public	どこからでも利用可能	そのカタマリー固有のモノ
#	Protected	当該クラス内とサブクラスから利用可能	（未使用）
-	private	当該クラス内からのみ利用可能	下位のカタマリーを踏襲したモノ
~	package	当該クラス内と、同一パッケージのクラスから利用可能	上位のカタマリーから具現化したモノ

テストカタマリー表記法：関連の表記

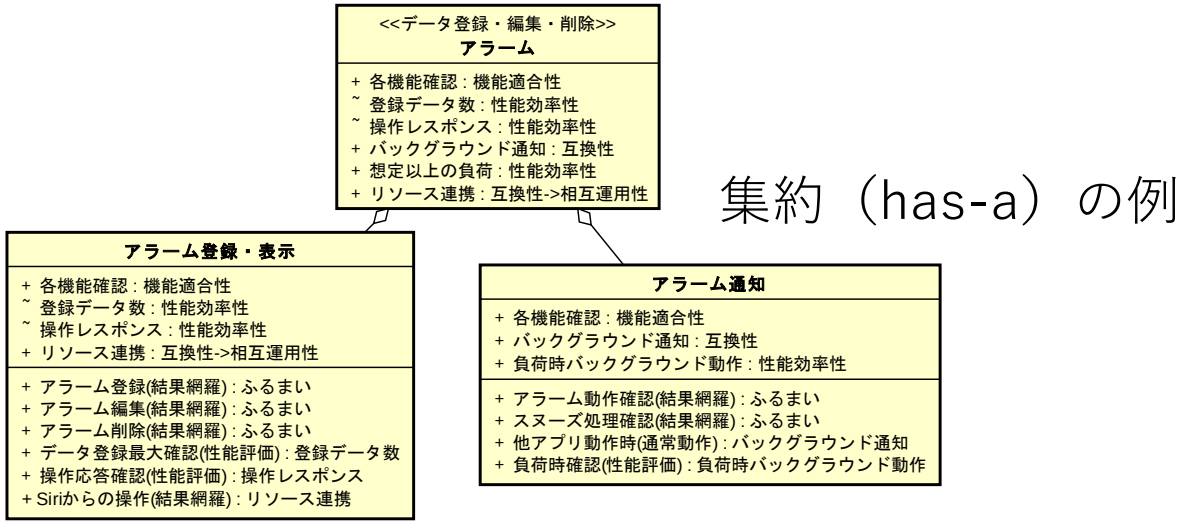
クラス図のとの違い

関係	線形	クラス図	カタマリー
関連 （association）		オブジェクト間の関連(あるいは関係)を表す。	（未使用）
集約 （aggregation）		「部分ー全体」の関係を表す。ひし形の方が全体を表わす。	カタマリー同士の関連線をつないでいる。機能の関連や実施手順の使用に対して関連するものを表す。
コンポジション （composition）		「部分ー全体」の関係を表す。ひし形の方が全体を表わす。部分が泡ラスオブジェクトが複数の全体を表すオブジェクトに共有されることがない。	（未使用）
依存 （dependency）		クラスのインスタンス間に利用関係がある事を表す。矢印の元側のクラスが利用者、先側のクラスが機能提供者。	（未使用）
汎化 （generalization）		継承関係を表す。異なるクラス間に類似性の性まとめた抽象クラスからの関連を表す。	パターンの継承を表す。
実現 （realization）		クラスが実現するうえで必要な共通操作のみを定義したインターフェースクラスとの関連を表す。	（未使用）

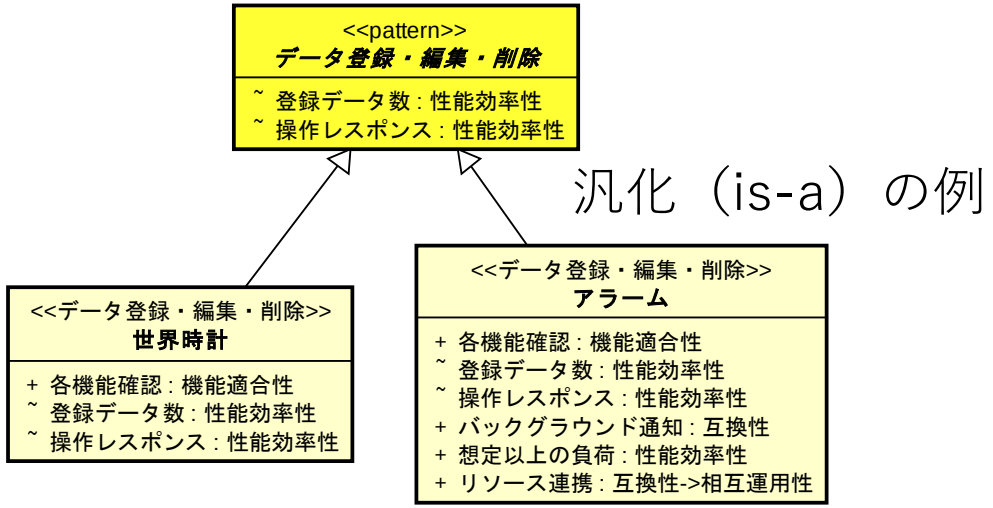
UMLクラス図と共通箇所

テストカタマリーはUMLツール（astah*やEnterprise Architect）のクラス図を使用してテストスイートモデルを表現できるように検討しております。
ただし、UML直接の表現とは異なる部分があります。

UMLと共通の部分：
　クラス間の関連（集約、汎化など）をそのまま使用できるようにしています。
　テストケースのグループ単位の関連性を表現しやすくしております。



集約（has-a）の例

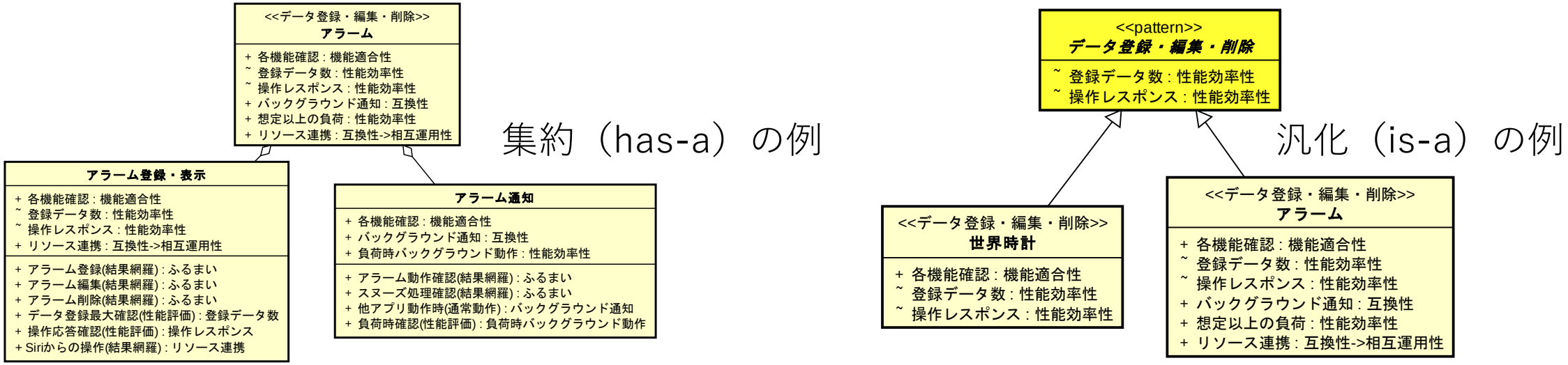


汎化（is-a）の例

テストカタマリー（UMLクラス図と共通箇所）

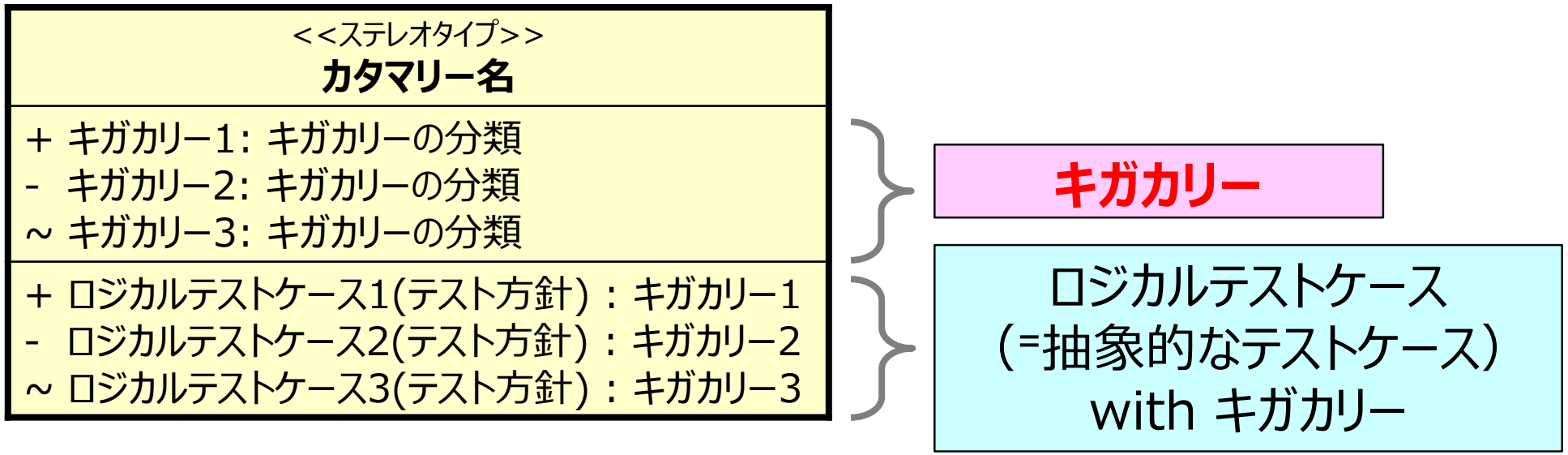
テストカタマリーはUMLツール（astah*やEnterprise Architect）のクラス図を使用してテストスイートモデルを表現できるように検討しております。
ただし、UML直接の表現とは異なる部分があります。

1．UMLと共通の部分：
　クラス間の関連（集約、汎化など）をそのまま使用できるようにしています。
　テストケースのグループ単位の関連性を表現しやすくしております。



テストカタマリーの構造

- テストカタマリーは複数のキガカリーを持つ
- ロジカルテストケースは１つのキガカリーと対応



おさらい：カタマリーで確認したい品質特性や気がかり、
テストケースの目的等をキガカリーと呼ぶ

テストカタマリーの一例

・「テストカタマリー」では1つの箱の中に
10、20ものテストケースを持たせたものを表現可能

アラーム登録・表示

+ 各機能確認：機能適合性
+ 登録データ数：性能効率性
+ 操作レスポンス：性能効率性
+ リソース連携：互換性->相互運用性

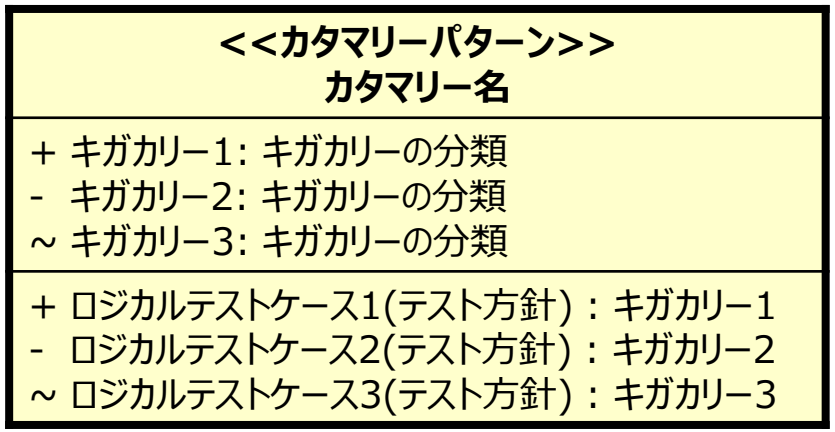
+ アラーム追加・個別編集、アラーム・スヌーズ()：各機能確認
+ アラーム追加・個別編集、曜日繰り返し設定()：各機能確認
+ アラーム追加・個別編集、ラベル()：各機能確認
+ アラーム編集()：各機能確認
+ アラーム削除()：各機能確認
+ データ登録最大数確認()：登録データ数
+ 操作応答確認()：操作レスポンス
+ Siriからの操作()：リソース連携
+ ライブラリ曲データ選択()：リソース連携

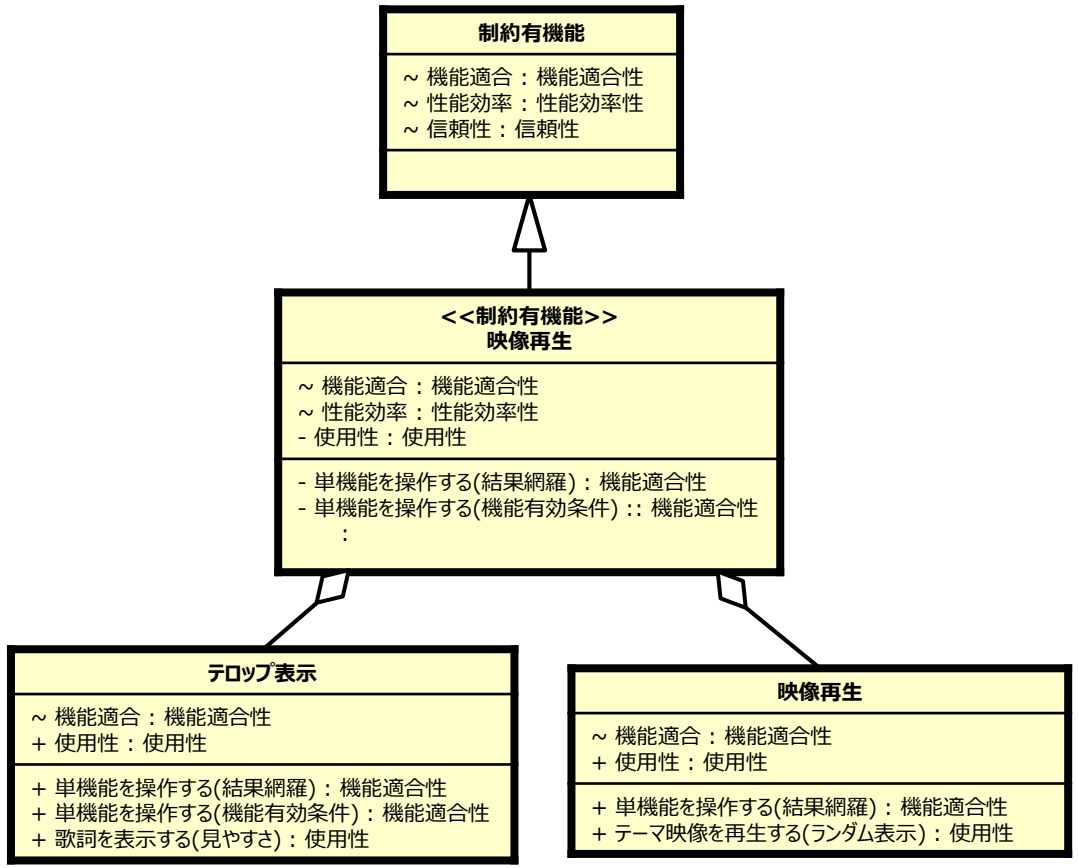
例：テストケース17件+パラメータ組合せに対応

カテゴリ/ID	項目名	概要	実施方法(案)	合否判定基準	ケース/設定	備考
1. アラーム						
1-1. アラーム登録・表示						
1-1-1 アラーム追加・個別編集、アラーム、スヌーズ						
1-1-1_001	アラーム時間設定の確認	アラームの時間を設定する	1. アラーム登録画面を開く 2. アラーム時間を設定する(時間設定はケース/設定に記載) 3.「保存」を押す	2. 設定した時間でアラーム登録がされていることを確認する	00:00 07:00 12:00 23:59	自動化対象
1-1-1_002	サウンド設定の確認	サウンド設定をする	1. アラーム登録画面を開く 2. サウンド設定画面を開く 3. サウンド設定をする 4.「保存」を押す	3. 設定したサウンド設定でアラーム登録がされていることを確認する ※ライブラリ曲に関しては、1-1-9で確認	デフォルト着信音を3件確認	自動化対象
1-1-1_003	スヌーズ設定の確認	スヌーズ設定をする	1. アラーム登録画面を開く 2. スヌーズのON/OFF設定をする 3.「保存」を押す	3. 設定したスヌーズ設定でアラーム登録がされていることを確認する	ON OFF	自動化対象
1-1-1_004	設定キャンセルの場合	操作中にキャンセルを実施して登録されないことを確認する	1. アラーム登録画面を開く 2. アラーム時間を設定する 3.「キャンセル」を押す	2. キャンセルが行われ、登録が行われないことを確認する	任意に福栖パターンの実施を推奨	自動化対象
1-1-2 アラーム追加・個別編集、曜日繰り返し設定						
1-1-2_001	曜日繰り返し設定の確認	曜日繰り返しの設定をする	1. アラーム登録画面を開く 2. 繰り返し設定画面を開く 3. 曜日繰り返し設定をする 4.「保存」を押す	3. 設定した曜日の繰り返し設定でアラーム登録がされていることを確認する	1-1-2_001_曜日設定表にまとめる	自動化対象
1-1-3 アラーム追加・個別編集、ラベル						
1-1-3_001	ラベル設定の確認	ラベル名称の設定をする	1. アラーム登録画面を開く 2. ラベル設定画面を開く 3. ラベル設定をする 4.「保存」を押す	3. 設定したラベル設定でアラーム登録がされていることを確認する	1-1-3_001_ラベル設定表にまとめる	自動化対象
1-1-4 アラーム編集						
1-1-4_001	編集選択での表示	選択したアラームが編集画面に移行できることを確認する	1. アラーム一覧画面を開く 2. 編集ボタンを押し、アラーム編集対象を選択する	2. アラーム編集対象の編集画面が表示されることを確認		設定の登録は1-1-1～1-1-3で実施可能
1-1-5 アラーム削除						
1-1-5_001	アラーム一覧の編集選択からの削除	アラーム一覧の編集ボタンを押した際に削除できることを確認する	1. アラーム一覧画面を開く 2. 編集ボタンを押し、アラーム編集対象「－」を押して削除を選択する	2. アラーム削除対象が削除されることを確認		自動化対象
1-1-5_002	編集画面からの削除	アラーム編集画面から削除できること	1. アラーム一覧画面を開く 2. 編集ボタンを押し、アラーム編集対象を選択する 3. 編集画面で削除ボタンを押す	3. アラーム削除対象が削除されることを確認		自動化対象
1-1-6 データ登録最大数確認						
1-1-6_001	データ登録数最大数確認	アラームデータの登録数の最大数を確認する	<アラーム登録> 1. アラーム登録画面を開く 2.「保存」を押してアラーム登録をする 3. <アラーム登録>を繰り返す	3. アラーム登録数が登録されていることを確認する	1-1-6_001_データ登録最大数表にまとめる	自動化対象
1-1-7 操作応答確認						
1-1-7_001	操作応答時間の確認	操作に対する応答時間を確認する	1-1-6_001の手順を実施	登録時のレスポンスが遅くなるようなケースがないこと	1-1-6_001と同じ	
1-1-8 Siriからの操作						
1-1-8_001	Siriによるアラーム追加	Siri経由でアラームを追加できることを確認する	1. Siriを呼び出す 2. 時間指定でアラームを追加する	2. 設定した時間でアラーム登録がされていることを確認する		
1-1-8_002	Siriによるアラーム削除 一覧呼び出し	Siri経由でアラームを削除の一覧を出せることを確認する	1. Siriを呼び出す 2. アラーム削除を呼び出す	2. 削除対象のアラーム一覧を表示することを確認する		
1-1-8_003	Siriによるアラーム削除 ラベルで削除	Siri経由でアラームを削除の一覧をからラベルでアラームを削除できることを確認する	1. Siriを呼び出す 2. アラーム削除を呼び出す 3. ラベルで削除対象を指摘する	3. 指定したラベルのアラーム削除がされていることを確認する		
1-1-8_004	Siriによるアラーム削除 時間で削除	Siri経由でアラームを削除の一覧をから登録時間でアラームを削除できることを確認する	1. Siriを呼び出す 2. アラーム削除を呼び出す 3. 時間で削除対象を指摘する	3. 指定した時間のアラーム削除がされていることを確認する		
1-1-9 ライブラリ曲データ選択						
1-1-9_001	着信音/通知音ストアから登録	着信音/通知音ストアから購入した曲をアラームとして登録できることを確認する	1. アラーム登録画面を開く 2. サウンド設定画面を開く 3. 着信音/通知音ストアからダウンロードして設定をする 4.「保存」を押す	4. 設定したサウンドの設定でアラーム登録がされていることを確認する		自動化対象
1-1-9_002	ライブラリ登録された曲から選択	保存されている曲ライブラリから曲をアラームとして登録できることを確認する	1. アラーム登録画面を開く 2. サウンド設定画面を開く 3. ライブラリから曲を選択して設定をする 4.「保存」を押す	4. 設定したサウンドの設定でアラーム登録がされていることを確認する		自動化対象

カタマリーの表現方法の例

カタマリーには、パターン化した抽象的なカタマリーとパターンを具現化した具象的なカタマリーがある。さらに具象化したカタマリーを細分化することができる。





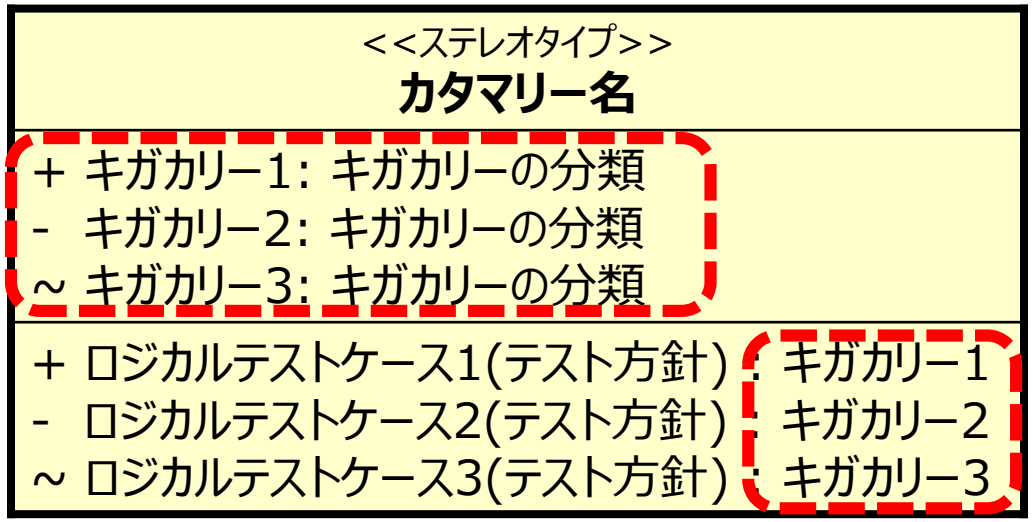
キガカリー＆ブンルイー

※キガカリー、ブンルイーは独自用語です

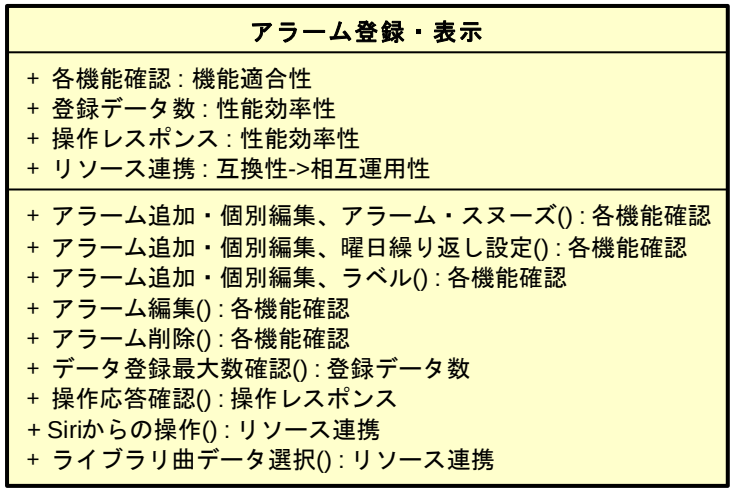
キガカリーについて

※キガカリーは独自用語です

- テストケースの持つ目的（意図）を明示する。
- テストの範囲を決めるため、抜けの確認に役立つ。
- 知識の蓄積やパターン化もできる。
- 「テスト観点（の、とある抽象度範囲）」もしくは「テストカテゴリ」と同等と考えてよい。



※実例



キガカリーの概念について

- テストケース/テストケースのグループは目的（意図）を持つ。
  - 目的のないテストケースは意味がない。

- テスト技法で導出されたテストケースも目的を持っている。
  - 同値クラス：結果に影響する入力値を確認する。
  - 境界値：不具合に繋がりやすい同値設定の境界部分を確認する。
  - DT/ドメイン分析：関係のある複数パラメータ組合せを確認する。

- 目的を実現するテストケースを明確にしたい。
  ※某テスト観点では、テストケースに割り当てる方法は明示していない。

- テストカタマリーではテストケースのグループが１つの目的に対応するように表現する。
  ※これはある意味仮説的、複数の目的に対応する場合もあるかもしれない。

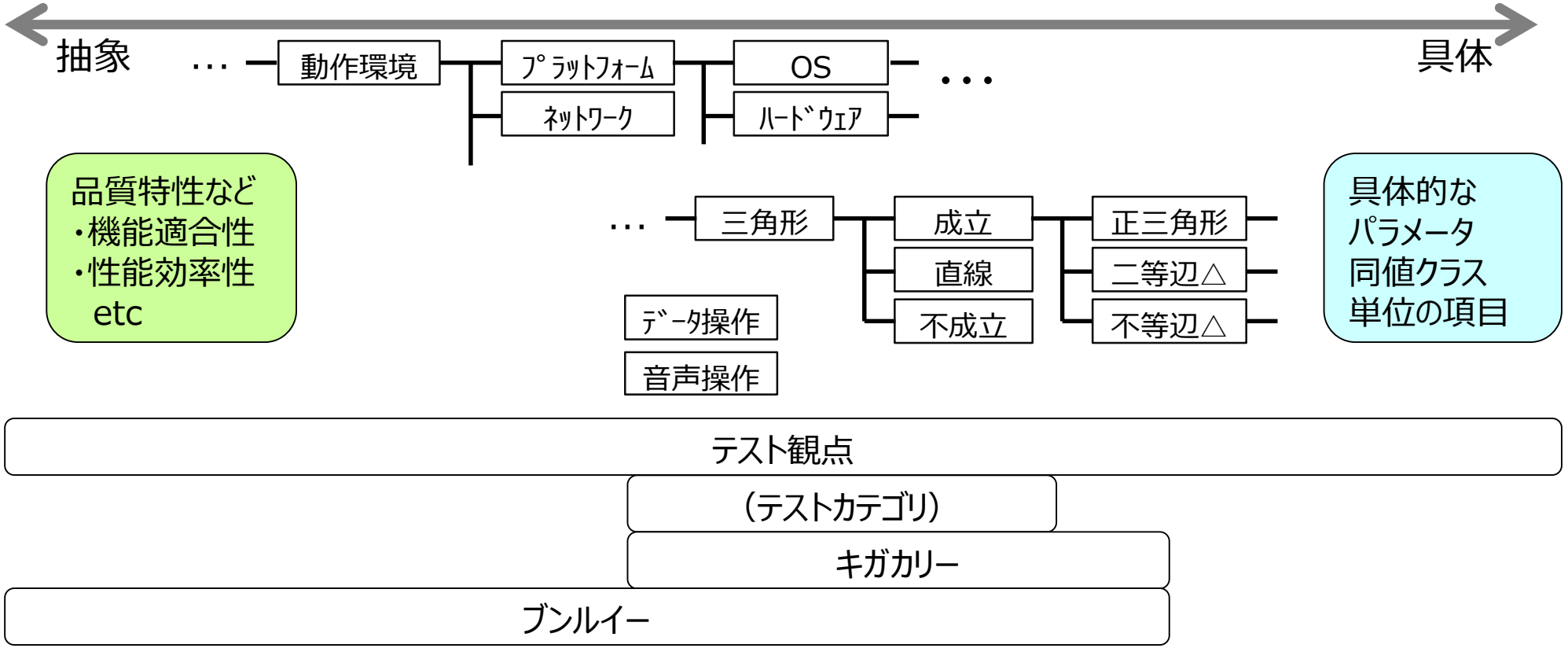
別の言葉で表してみる

キガカリーを別の言葉で表すと、次のようなものとなりそう。

- 保証すべきもののキーワード
- 確認したいもの
- テストの関心ごと
- テストケースの目的/意図
- 気がかり事項/リスク

≒ テスト観点上のとある抽象レベル
≒ テストカテゴリ

テスト観点の範囲に対する概念MAP



他手法との関連

ゆもつよメソッド

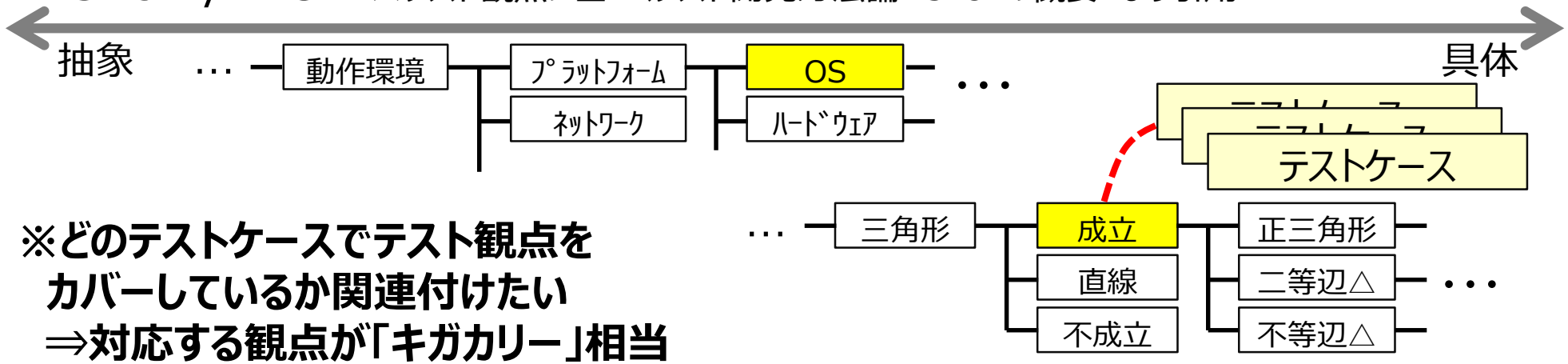
テストカテゴリ

機能名	機能小分類		データ操作	音声操作	…
検索	キュー		1件	1件	
	データ送り		2件	-	
再生	1件再生		1件	5件	
・・・					

データ送り × データ操作
1件再生 × データ操作
1件再生 × 音声操作

キガカリーに相当

VSTeP / NGT ※テスト観点に基づくテスト開発方法論VSTePの概要　より引用



※どのテストケースでテスト観点を
カバーしているか関連付けたい
⇒対応する観点が「キガカリー」相当

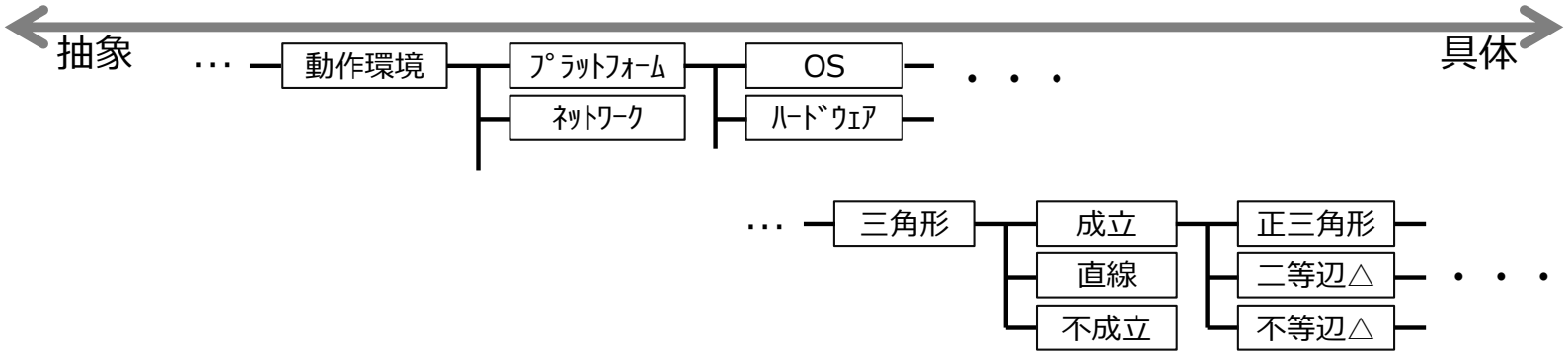
他手法との関連

※把握・理解している内容で説明
・ゆもつよメソッド：
テストケースはいずれかの
テストカテゴリに属する。
テストカテゴリが、
そのテストケースの目的（意図）を示す。

テストカテゴリ

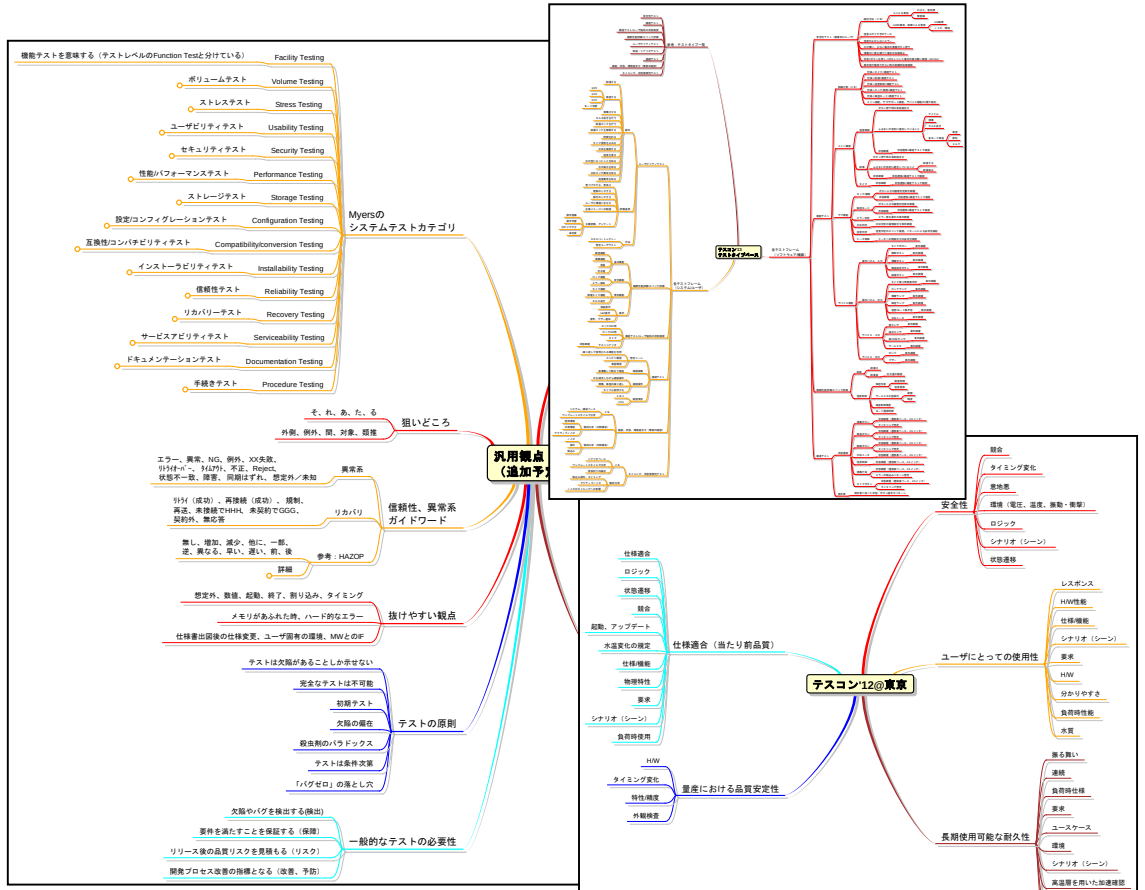
機能名	機能小分類		データ操作	音声操作	…
検索	キュー		1件	1件	
	データ送り		2件	-	
再生	1件再生		1件	5件	
・・・					

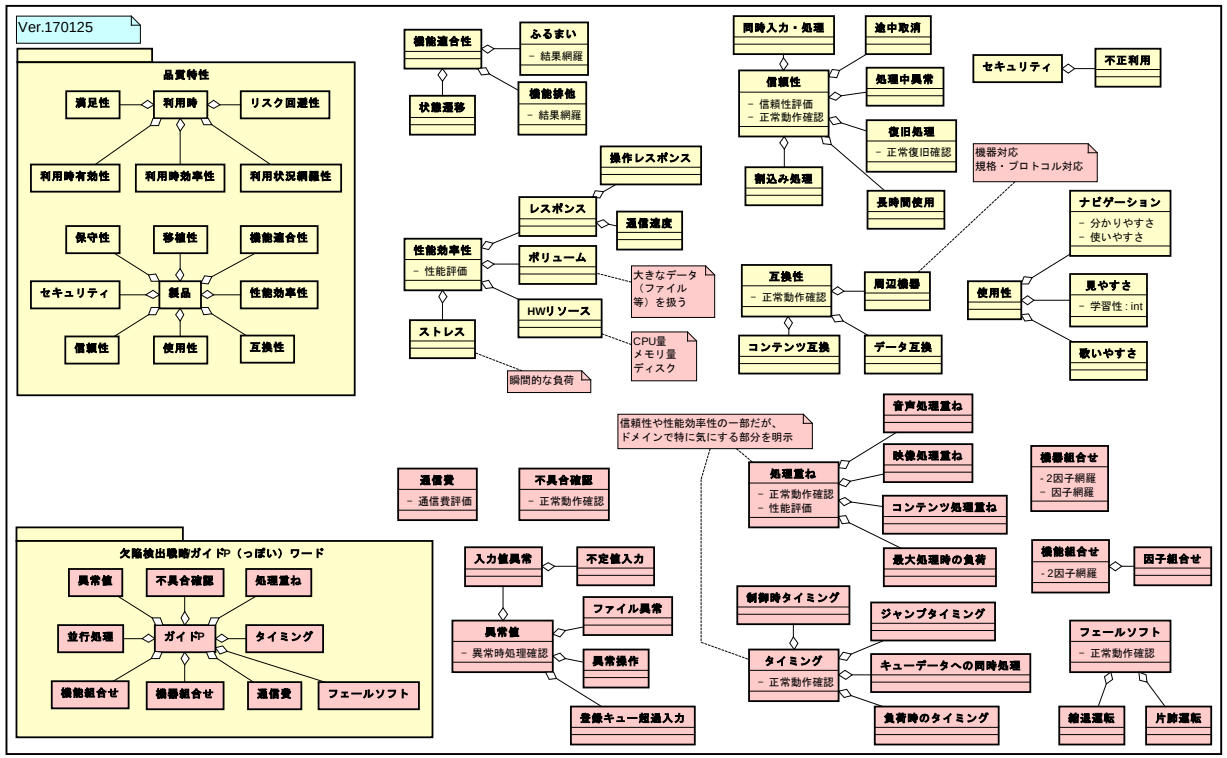
・VSTeP：テストケースの目的（意図）をモデリングしている。
ただし、テストケースに割り当てる方法を明示はしていない。



ブンルイー

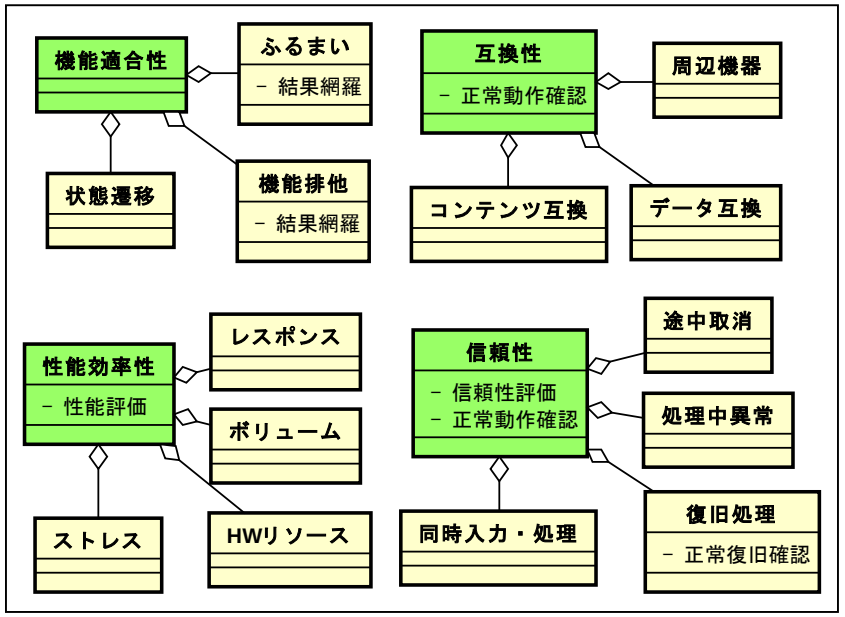
キガカリーは蓄積することで役立つ情報となります。
整理して全体を示したものを「ブンルイー」と呼ぶ。 ※独自用語です



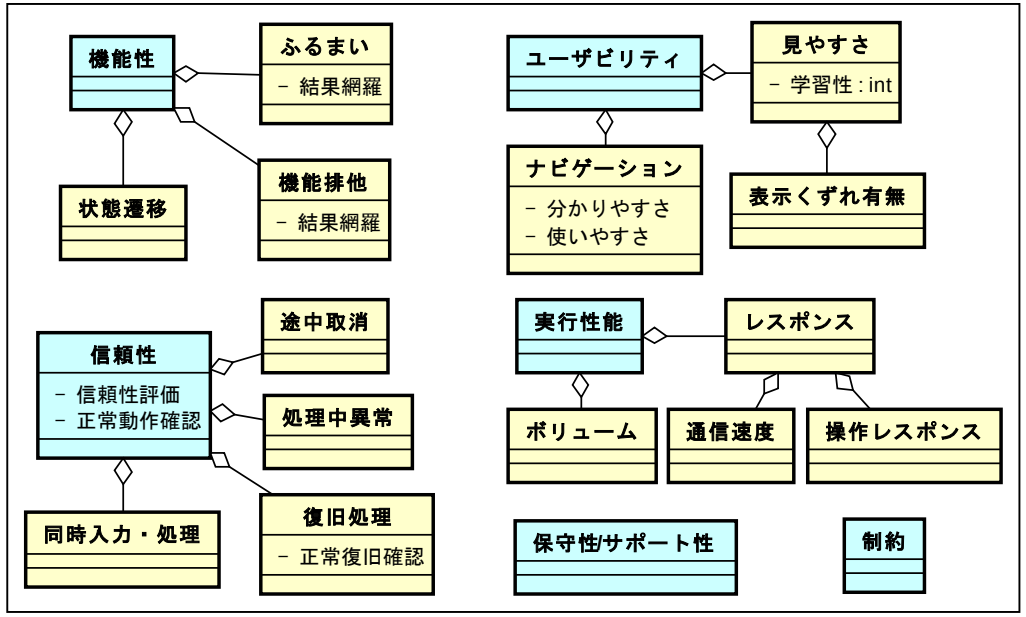


作り方には決まりはない

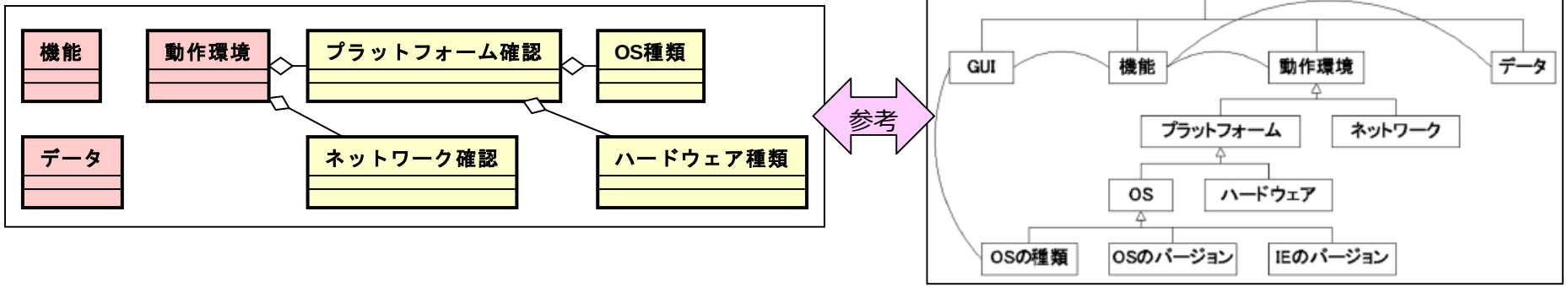
<品質特性ベース>



<FURPS+ベース>



<VSTeP形式> ※ボトムアップに構築になるかも



テスト方針（ポリシー）

よく出てくるテスト方針

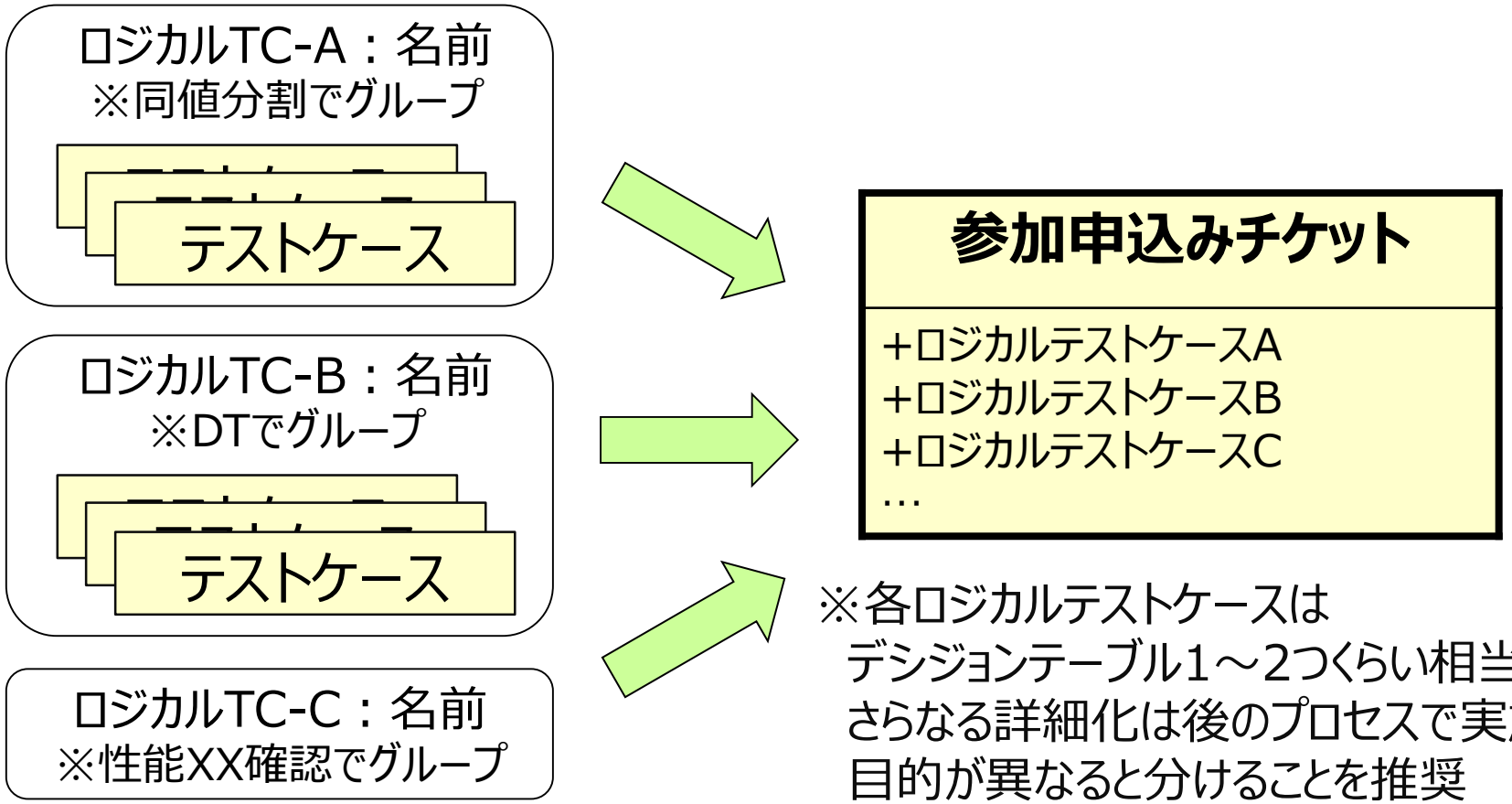
- 結果網羅
- 機能有効条件
- 見やすさ
- タイミング
- 性能評価

上記のような「網羅すべき、確認すべき基準」を記載します。

カタマリー例

テストカタマリー推奨のテスト粒度

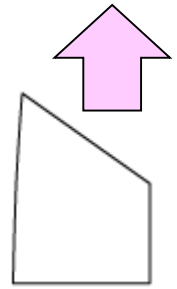
テストカタマリーは「ロジカルテストケース」※特にレベル1～2
相当の抽象度のテストケースを持つことを推奨します。



テストカタマリーの概要

アラーム登録
+各機能　：機能特性 ~登録数　：性能効率性 …
+アラーム登録：各機能 +アラーム解除：各機能 ~最大登録数検証：登録数 …

iOS時計アプリ
アラーム登録



詳細

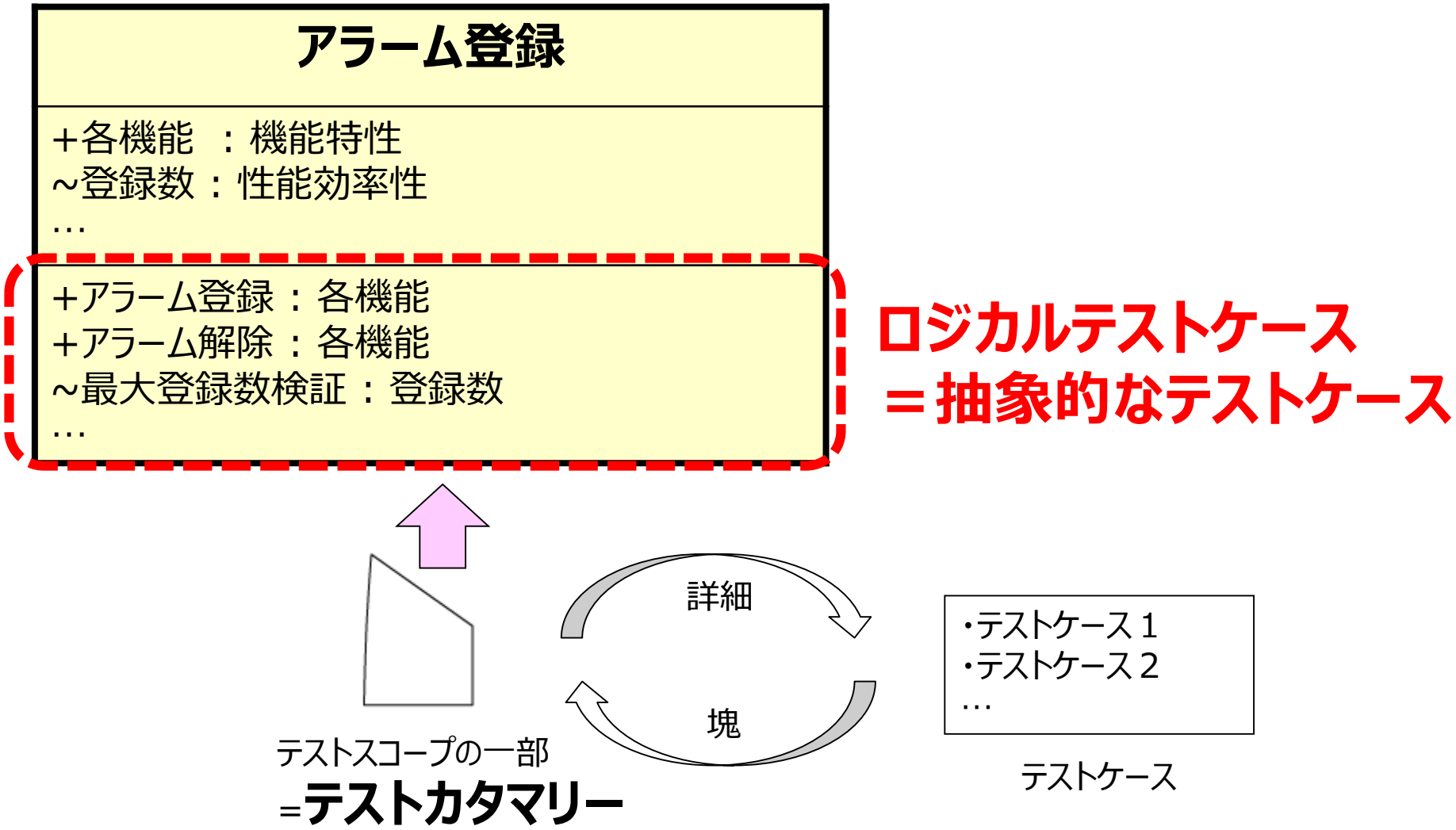
塊

テストスコープの一部
=**テストカタマリー**

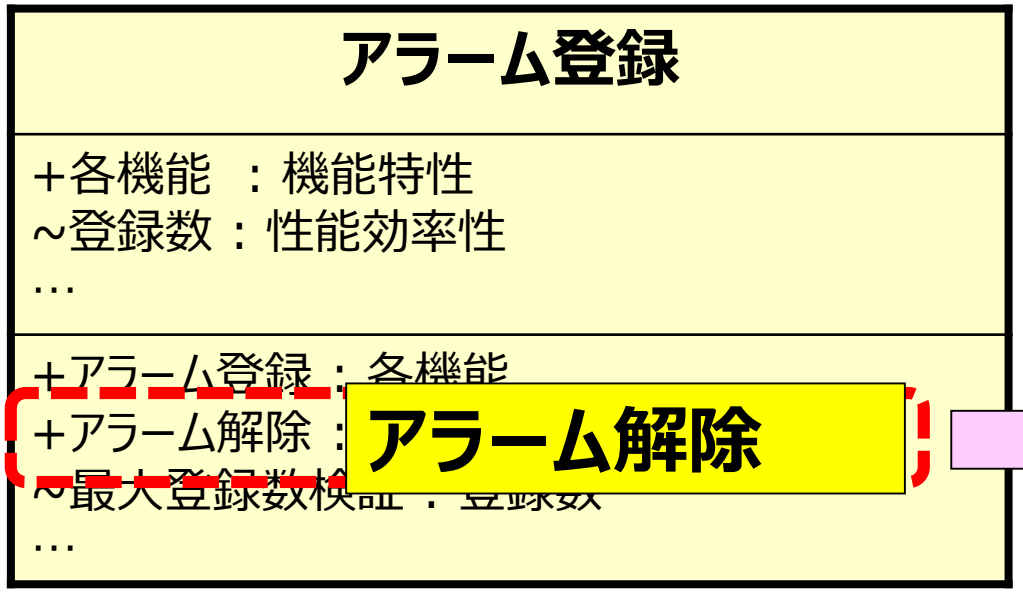
・テストケース１
・テストケース２
…

テストケース

テストカタマリーの概要



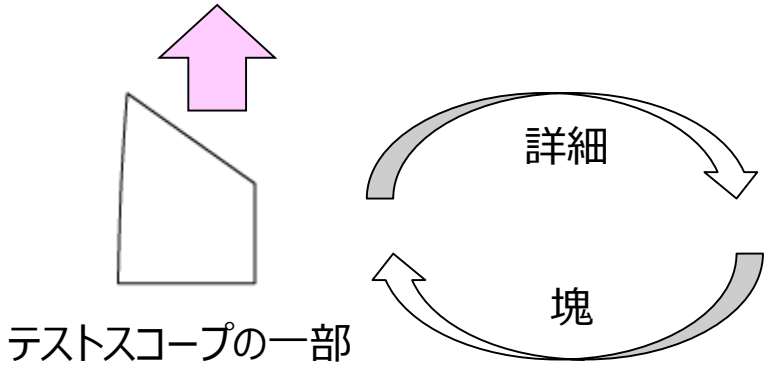
テストカタマリーの概要



各ロジカルテストケースから
具体的なテストケースを作成
※テスト詳細設計で実施

複数の具体的テストケース

編集画面にて登録解除
アラーム画面にて登録解除
未登録のときに登録解除



テストスコープの一部
=**テストカタマリー**

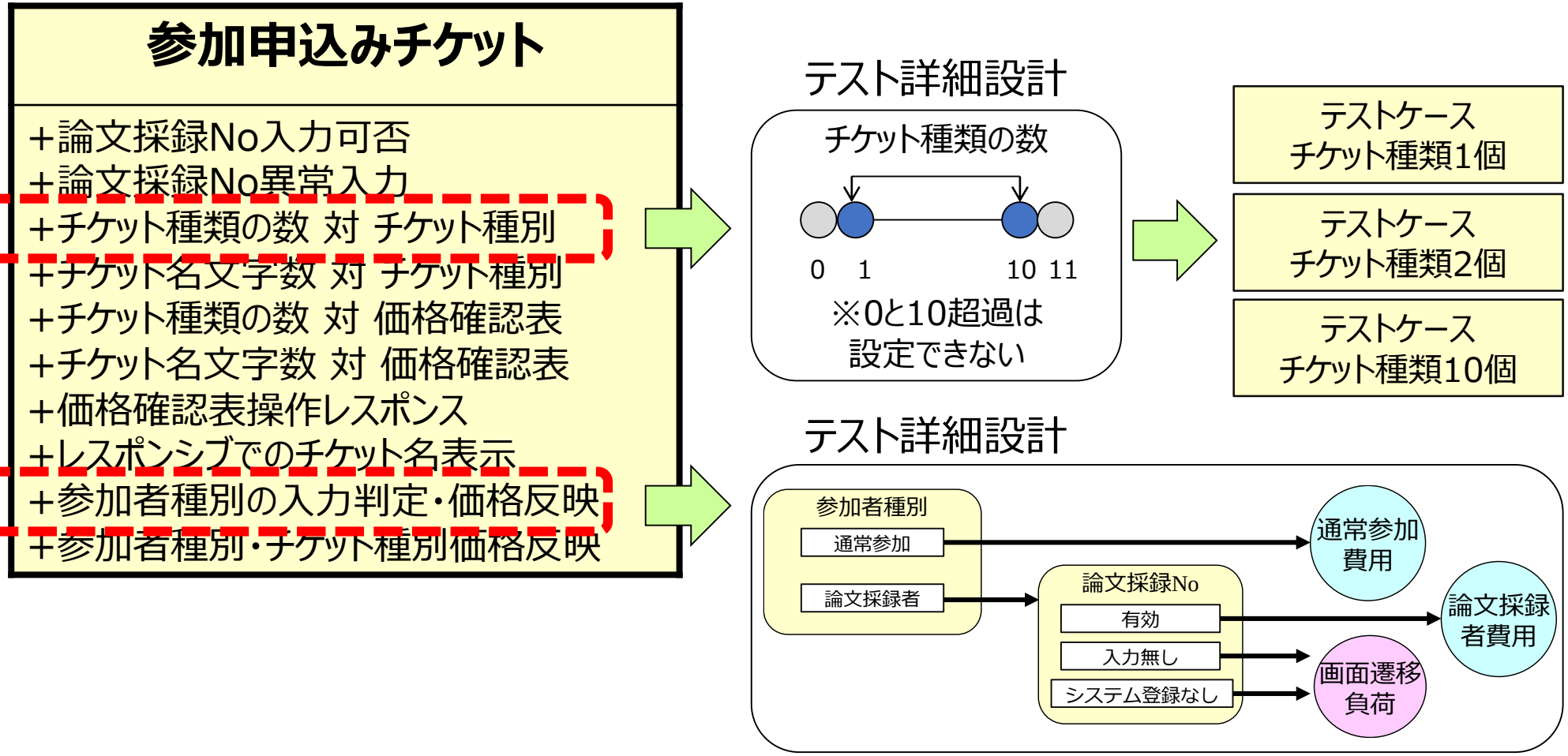
・テストケース１
・テストケース２
…

テストケース

カタマリーノウハウ

ロジカルテストケースとテストケース

上位から考える場合には、ロジカルテストケースまで一度考えた後に、テスト詳細設計で具体的テストケースをつくります。



テストケース仕様書を作る場合

テストケース仕様書を作成する場合や、テスト管理ツールにて
テストケースを管理する場合に階層構造はシンプルに対応してます。

参加申込みチケット

＋論文採録No入力可否
＋論文採録No異常入力
＋チケット種類の数 対 チケット種別
＋チケット名文字数 対 チケット種別
＋チケット種類の数 対 価格確認表
＋チケット名文字数 対 価格確認表
＋価格確認表操作レスポンス
＋レスポンシブでのチケット名表示
＋参加者種別の入力判定・価格反映
＋参加者種別 チケット種別価格反映

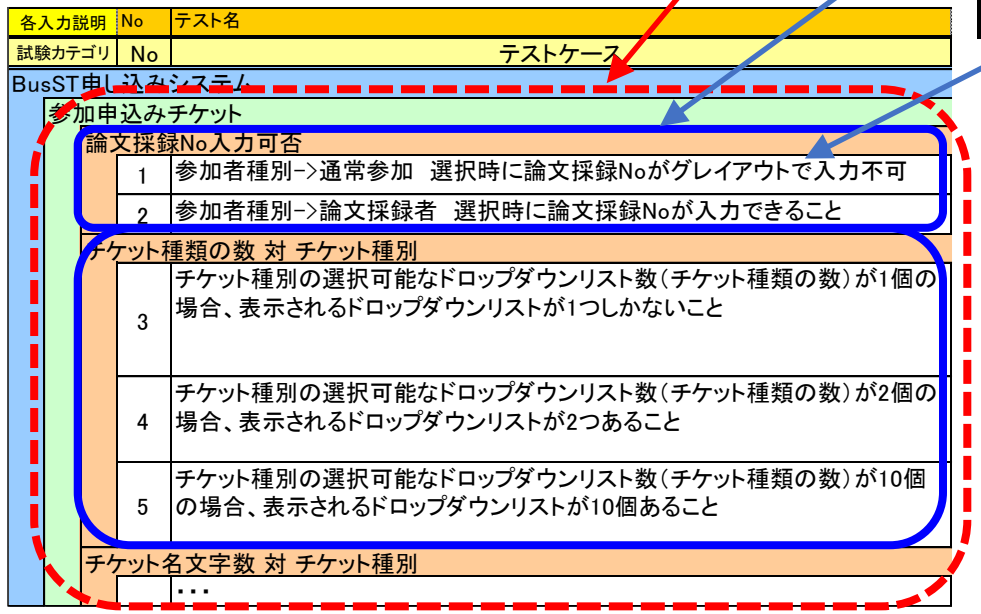
ロジカルテストケースの
テストケース範囲は赤点線

テストカタマリーのもつ
テストケースは青点線範囲

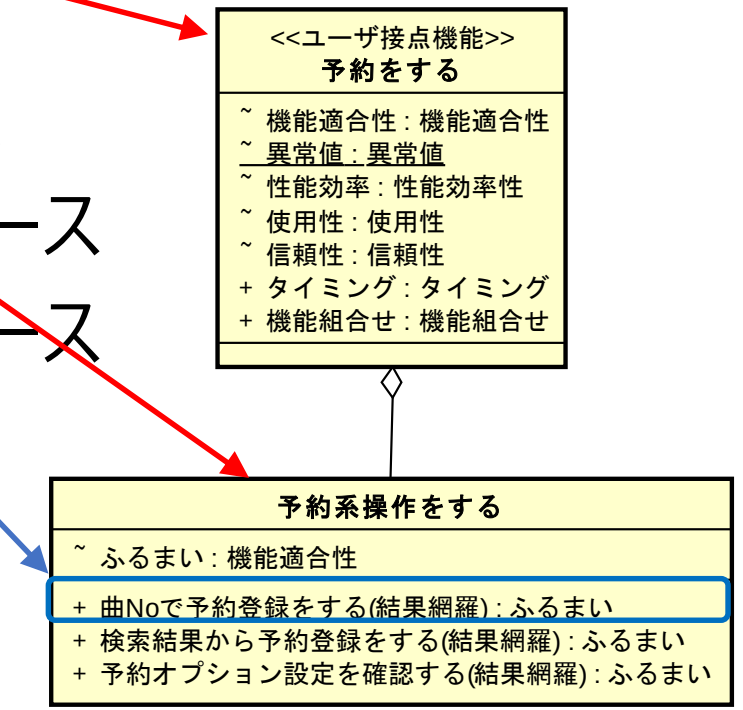
各入力説明	No	テスト名
試験カテゴリ	No	テスト
BusST申し込みシステム		
参加申込みチケット		
論文採録No入力可否		
	1	参加者種別->通常参加　選択時に論文採録Noがグレイアウトで入力不可
	2	参加者種別->論文採録者　選択時に論文採録Noが入力できること
チケット種類の数 対 チケット種別		
	3	チケット種別の選択可能なドロップダウンリスト数（チケット種類の数）が1個の場合、表示されるドロップダウンリストが1つしかないこと
	4	チケット種別の選択可能なドロップダウンリスト数（チケット種類の数）が2個の場合、表示されるドロップダウンリストが2つあること
	5	チケット種別の選択可能なドロップダウンリスト数（チケット種類の数）が10個の場合、表示されるドロップダウンリストが10個あること
チケット名文字数 対 チケット種別		
		・・・

カタマリーとレベルテストケースの関係

カタマリー
- カタマリー
  - ロジカルテストケース
    - ハイレベルテストケース
      - ローレベルテストケース
      - ローレベルテストケース

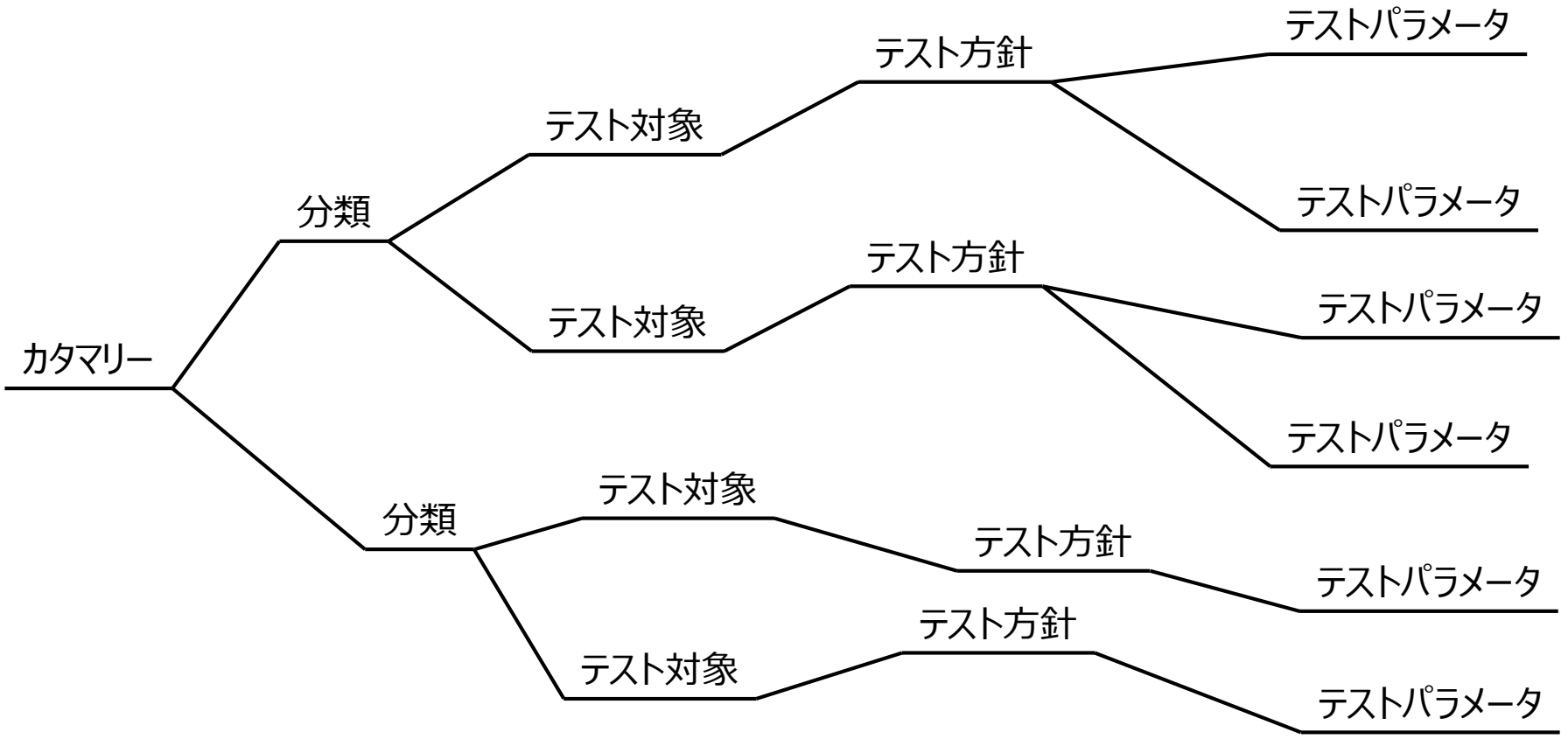


各入力説明	No	テスト名
試験カテゴリ	No	テストケース
BusST申し込みシステム		
参加申込みチケット		
論文採録No入力可否		
	1	参加者種別->通常参加　選択時に論文採録Noがグレイアウトで入力不可
	2	参加者種別->論文採録者　選択時に論文採録Noが入力できること
チケット種類の数 対 チケット種別		
	3	チケット種別の選択可能なドロップダウンリスト数（チケット種類の数）が1個の場合、表示されるドロップダウンリストが1つしかないこと
	4	チケット種別の選択可能なドロップダウンリスト数（チケット種類の数）が2個の場合、表示されるドロップダウンリストが2つあること
	5	チケット種別の選択可能なドロップダウンリスト数（チケット種類の数）が10個の場合、表示されるドロップダウンリストが10個あること
チケット名文字数 対 チケット種別		
	...	



小さいカタマリー

小さいカタマリーを考えるの場合、ツリー状にテストケースを下記のように考えると良いです。
※テストケース出しの段階では下記の形を意識する必要はない。



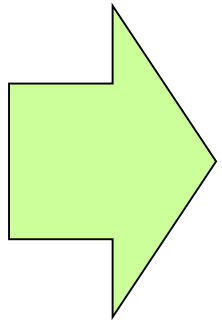
カタマリーが大きすぎる場合

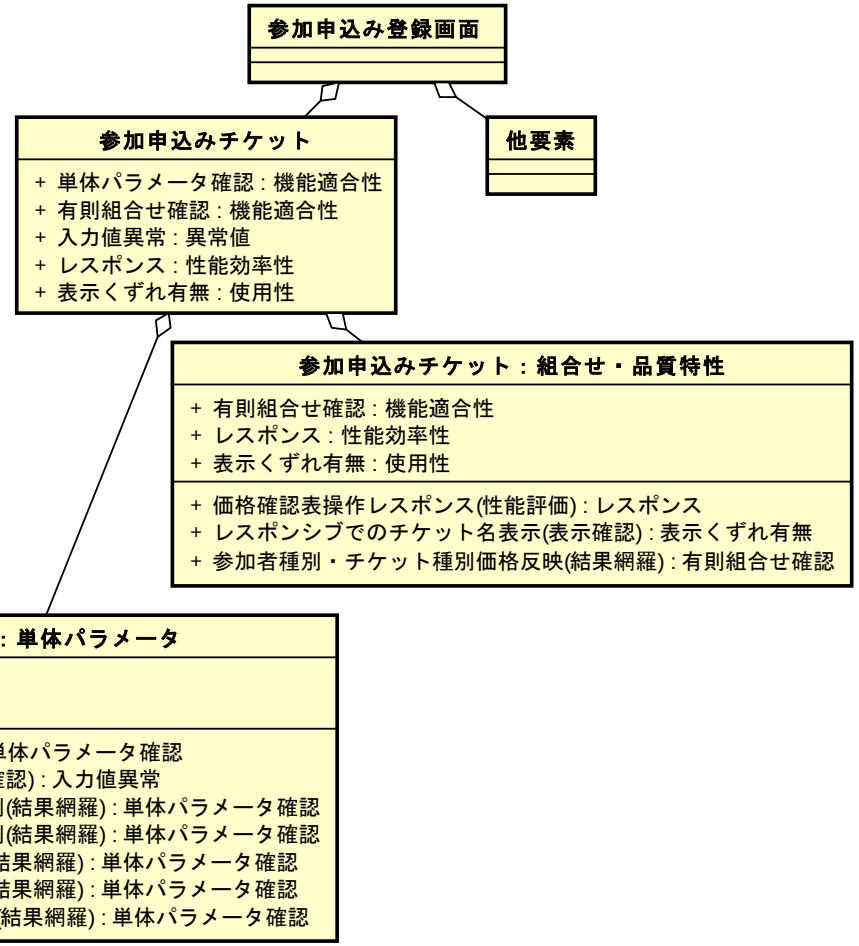
１つのカタマリーが大きすぎる場合、分割を推奨。
「has-a（集約）」を用いて分割を明示。

参加申込みチケット

+ 単体パラメータ確認 : 機能適合性
+ 有則組合せ確認 : 機能適合性
+ 入力値異常 : 異常値
+ レスポンス : 性能効率性
+ 表示くずれ有無 : 使用性

+ 論文採録No入力可否(結果網羅) : 単体パラメータ確認
+ 論文採録No異常入力(異常時処理確認) : 入力値異常
+ チケット種類の数 対 チケット種別(結果網羅) : 単体パラメータ確認
+ チケット名文字数 対 チケット種別(結果網羅) : 単体パラメータ確認
+ チケット種類の数 対 価格確認表(結果網羅) : 単体パラメータ確認
+ チケット名文字数 対 価格確認表(結果網羅) : 単体パラメータ確認
+ 価格確認表操作レスポンス(性能評価) : レスポンス
+ レスポンシブでのチケット名表示(表示確認) : 表示くずれ有無
+ 参加者種別の入力判定・価格反映(結果網羅) : 単体パラメータ確認
+ 参加者種別・チケット種別価格反映(結果網羅) : 有則組合せ確認





書き方の一例

ロジカルテストケース

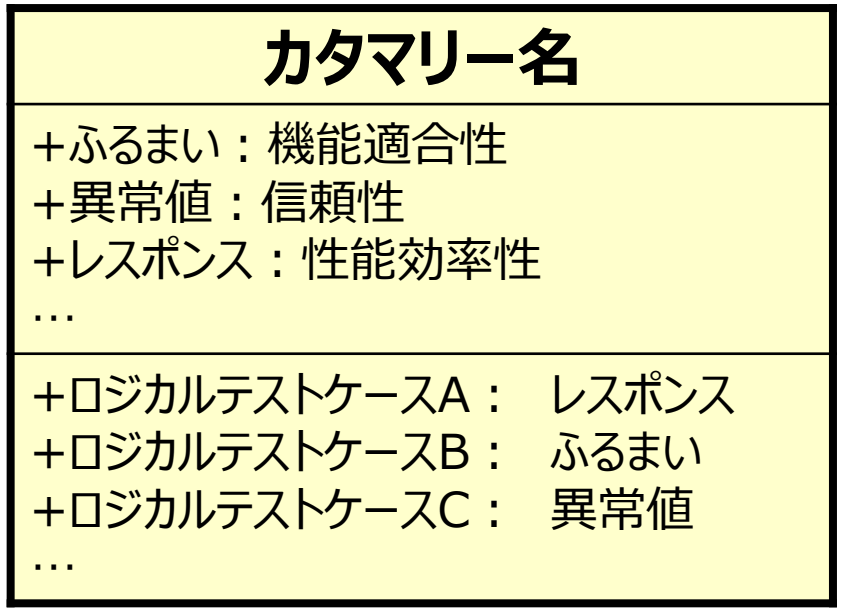
「○○を××する」という感じに整理。
「××」が「確認」、「評価」というのが出てきたらそれは方針のことを言っていたり、対象と方針を一緒くたにしている可能性がある。

テスト方針

「○○を確認する」「○○を評価する」の「○○」。
「結果網羅を確認する」という場合は、「結果網羅」がテスト方針の箇所に当たる。

記法を用いるにあたって

- 全記法を必ず使用する必要はありません。
- クラス図の記法と異なる使用がありますのでご注意を。
  ※今後の課題、変わるかもしれません。



参考情報

用語

テストスイートアーキテクチャ

段階的に設計されたテストケースで構成されるテストスイートの構造。
ソフトウェアの内部に焦点を当てている。

テストシステムアーキテクチャ

シミュレータなどテスト実行環境のシステム構成に焦点を当てている。

用語

テストカタマリー
　テストケースの集合体（塊）を表現したもの
　「カタマリー」と略して呼ぶ場合もある
　テストカタマリー記法自体を「テストカタマリー」と呼ぶ場合もある

キガカリー
　カタマリーが確認したい品質特性や気がかり、テストケースの目的等

ブンルイー
　上記キガカリーを分類し、特定のプロジェクトや組織単位で使う一覧としたもの
　ツリー構造やネットワーク構造で整理したもの。ブンルイー図と呼ぶ場合もある

ブログ情報

- テストケースの表現と粒度
  http://blog.amateur-factory.jp/?eid=1444287
- テストケースの表現と粒度（マニアック編～ISTQB（JSTQB）の情報から考える）
  http://blog.amateur-factory.jp/?eid=1444288

ブログ情報

- テストカタマリーの紹介１：カタマリーの概要
  http://blog.amateur-factory.jp/?eid=1444271
- テストカタマリーの紹介２：カタマリーの例
  http://blog.amateur-factory.jp/?eid=1444272
- テストカタマリーの紹介３：カタマリーの記法とツール（前編）
  http://blog.amateur-factory.jp/?eid=1444273
- テストカタマリーの紹介４：カタマリーの記法とツール（後編）
  http://blog.amateur-factory.jp/?eid=1444274
- テストカタマリーの紹介５：カタマリーパターンの活用
  http://blog.amateur-factory.jp/?eid=1444275
- テストカタマリーの紹介６：まとめ
  http://blog.amateur-factory.jp/?eid=1444276

レポート他

- OSC2017 in Hokkaido
  「もしかして、テストって誰でもできると思っちゃってませんか？ 最新のテストの動向を知りたくないですか？」
  https://www.slideshare.net/tef-do/tefdo-osc2017-in-hokkaido-07152017
- テスト設計コンテスト'17 OPENクラス 決勝戦レポート
  http://aster.or.jp/business/contest/contest2017.html
- 公開用 テストカタマリーワークショップ（説明のみ）

  https://www.slideshare.net/NoriyukiMizuno/ss-80257236

参考論文

- OMG, “UML Testing Profile (UTP) Version 2.0 - Beta,” http://www.omg.org/spec/UTP/2.0/, September 2017.
- OMG, “UML Version 2.4.1,” http://www.uml.org/, 2012.
- OMG, “SysML Version 1.5,” http://www.omgsysml.org/, 2017.
- ISO/IEC 25010:2011, “Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE)−System and software quality models,” 2011.

参考論文

- Glenford J. Myers, Tom Badgett, Corey Sandler, "The Art of Software Testing 3rd edition," Wiley, 2011.
- Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides, “Design Patterns: Elements of Reusable Object-Oriented Software,” Addison-Wesley Professional, 1994
- Y.Nishi, “Design principles in Test Suite Architecture,” International Workshop on Software Test Architecture (InSTA 2015), Graz, Austria, April 2015.
- Y. Nishi, T. Katayama and S. Yoshizawa, “Combinatorial Test Architecture Design Using Viewpoint diagram,” International Workshop on Combinatorial Testing (IWCT2013), Luxembourg, March 2013, CD-ROM.

参考論文

- T.Yumoto, T.Matsuodani, and K.Tsuda, ” A Study on the effectiveness of Test-Categories based test analysis” , International Workshop on Software Test Architecture (InSTA2016), Chicago, April 2016.
- N.Mizuno, M.Nakakuni and Y.Seino, “Test Conglomeration - Proposal for Test Design Notation like Class Diagram,” International Workshop on Software Test Architecture (InSTA2017), March 2017. http://ieeexplore.ieee.org/document/7899074/
- M.Nakakuni, S.Ban, N.Mizuno, K.Abe, H.Iseri, T.Odan, H.Ando, K.Iseri, A.Satoh “Proposal for Enhancing UTP2 with Test Aspects,” International Workshop on Software Test Architecture (InSTA2018), April 2018.