テスト設計の本質を改めて考えてみる

～生成AIを活用する時代だからこそ、作ったテストの説明性を高めよう～

2026年7月3日(金)

於 JaSST' 26 Kansai

山﨑 崇

株式会社ベリサーブ 品質保証部プロジェクト推進課 課長 兼
ソフトウェア品質コンサルティング部 シニアコンサルタント

2001年からセキュリティ対策ベンダーにおいて、さまざまなプロジェクトのソフトウェアテスト活動にQAエンジニアとして従事。その傍ら、さまざまなテストコミュニティに参画し、活動の場を広げる。2015年にベリサーブに入社。現場への技術支援、教育、コンサルテーションなどを担い、少しでも現場が幸せになるように日々奮闘中。

主な活動

- SQuBOKガイドV4 策定部会(’25～)
- 日科技連ODC分析研究会 運営委員(’24～)
- JaSST東京実行委員会 (’11～23)
- テスト設計コンテストU-30審査委員 (’16～)
- ASTER テストプロセス改善研究会 (’16～)
- JSTQB Foundation Level 認定講師 (’12～)など

主な著書・訳書

- 実践ソフトウェアエンジニアリング 第9版 (共訳)
- 開発現場で役立つテスト「超」実践講座 第9回 (著)
- テスト技術者資格制度Foundation Level Extension シラバス アジャイルテスト担当者 (共訳)
- システム開発ジャーナル Vol. 10 (共著)
- ソフトウェア・テストPRESS Vol. 8(共著)など

主な講演

- SQiPシンポジウム2024併設チュートリアル「テスト戦略」
- JaSST’22四国「テストを学び成長する 2022年度版」
- JaSST’21北陸 基調講演「テストを学び成長する」
- JaSST’17東北 基調講演「テストの極みを目指して」
- JaSST’15東京 チュートリアル「初心者からの脱出」
- テスト設計コンテストチュートリアル U-30クラス(’17～’19)ちびこん編(’21)
- SQiP研究会 品質保証の基礎コース 第7回「ソフトウェアテストの概観を識る」(’17, ’20～)
- 第14回SPIトワイライトフォーラム「テストプロセス改善モデルの最新動向」など

その他 https://www.linkedin.com/in/yamasaki696/ を参照

特技ソフトウェアテスト

趣味ソフトウェアテスト

Yes, I’m loving Software Testing!

@yamasaki696

この講演のゴール

テスト設計について改めて考えてみて、

テストベースから説明可能なテストケースを

導く考え方の足掛かりを得る

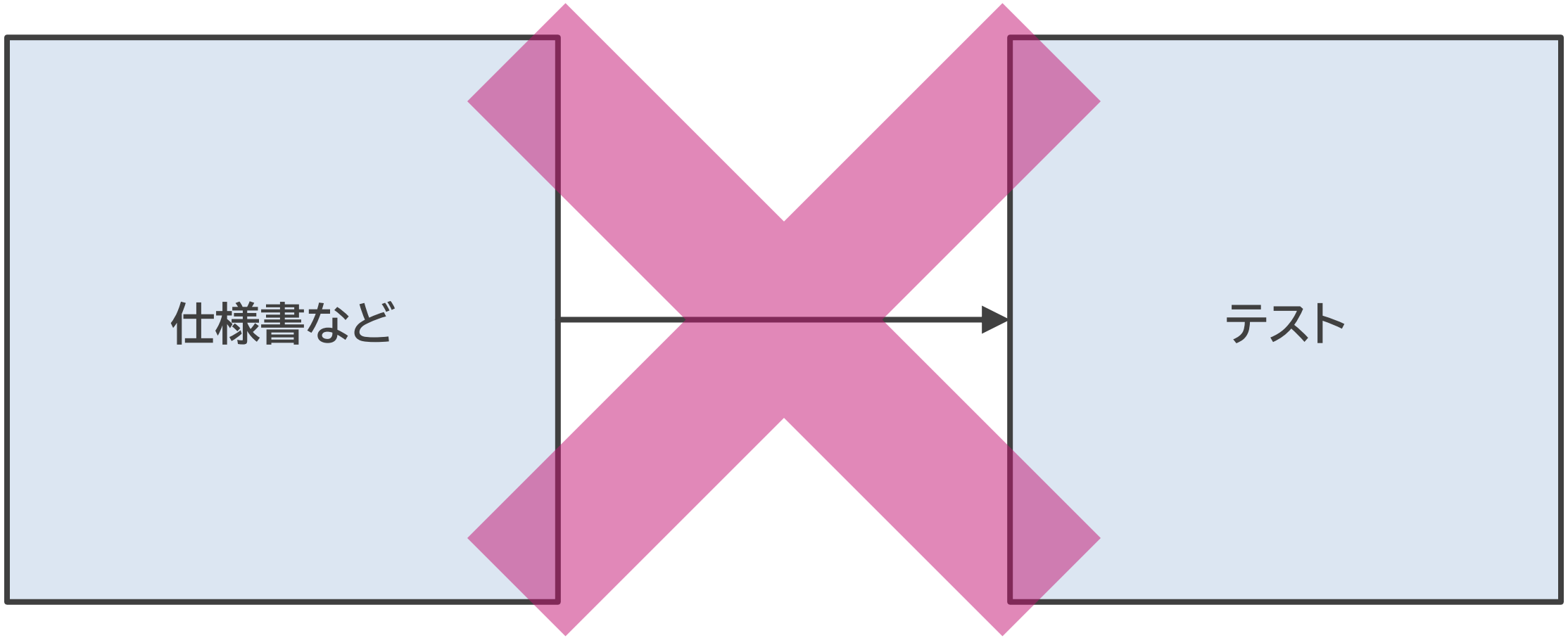
良く聴くお悩み

テスト技法を勉強して演習ではうまく
使えたのに実務ではどうやって技法を
適用すれば良いのか分からずうまく
使えていません…🥺

良く聴くお悩み

テストベースを生成AIに与えて
テストケースを作ってもらったら
それっぽいテストケースができたけど、
それで大丈夫なのか心配です…🥺

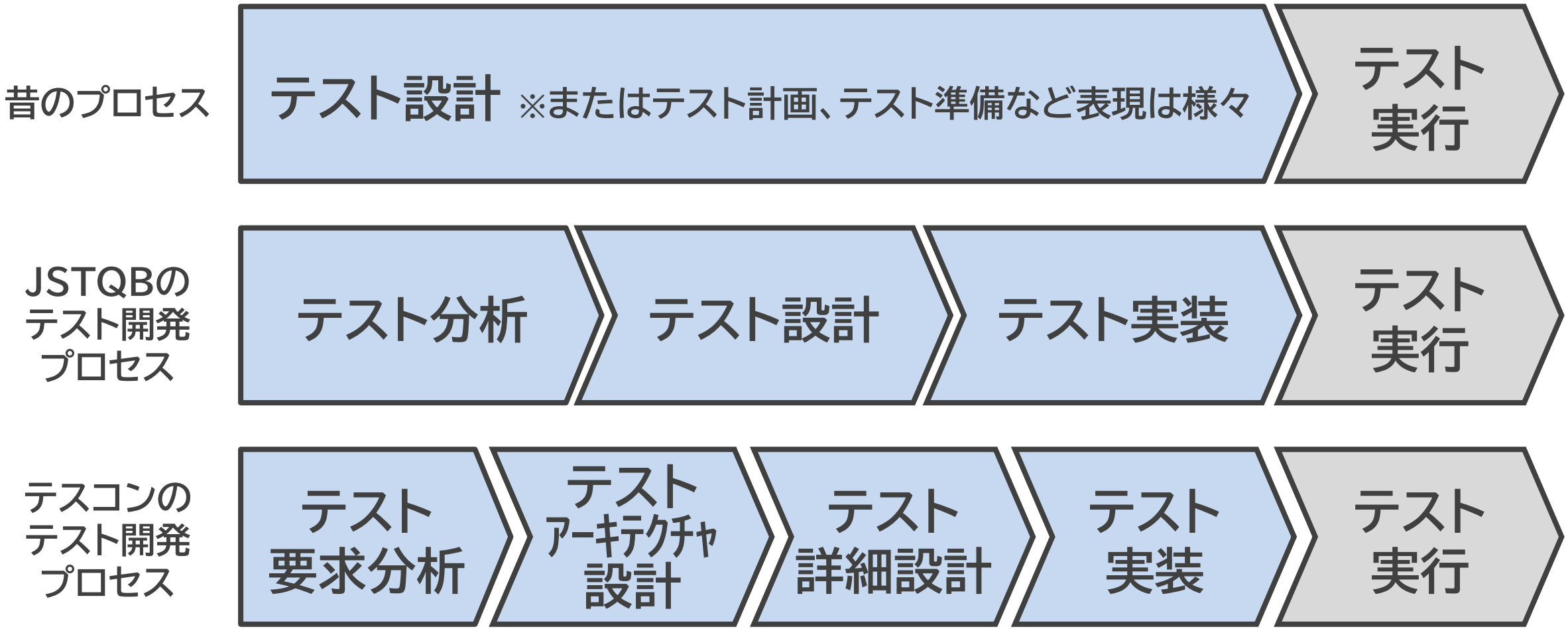
テストベースから直接テストを作ろうとしている？



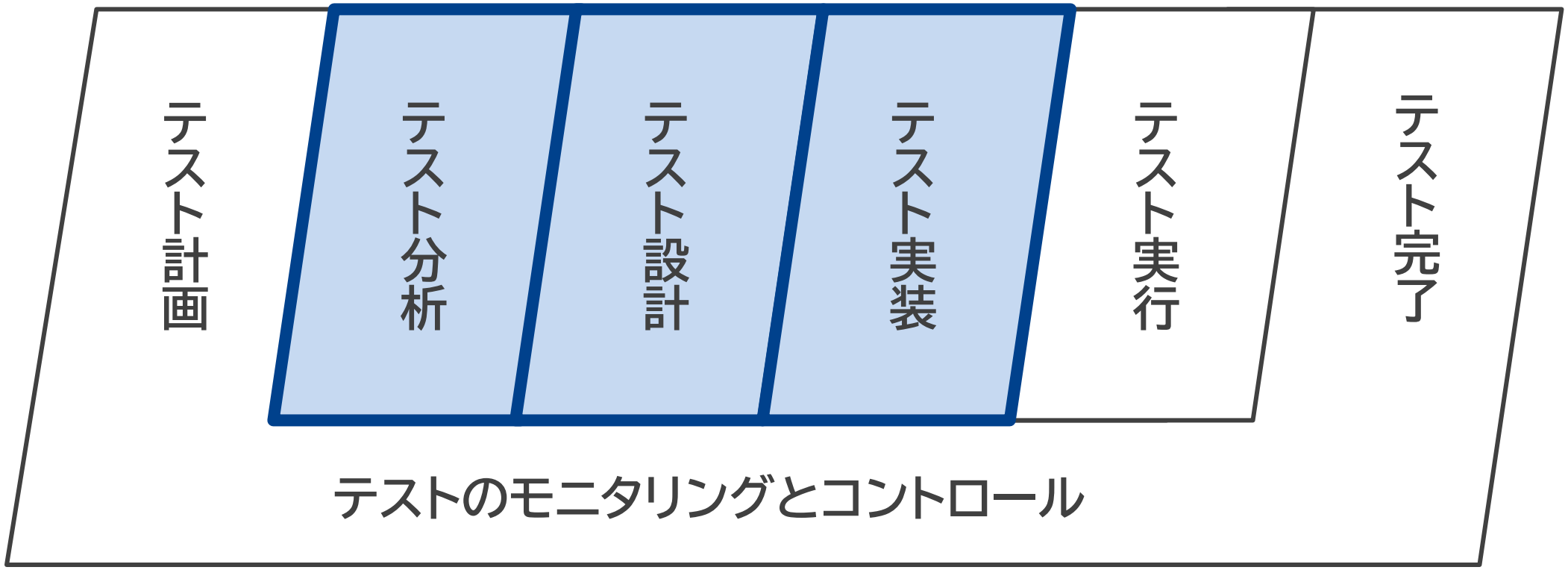
納得感の高いテストを作るには、いきなり個別のテストケースを作っても駄目

テストを段階的に詳細化して、説明性の高いテストを作り、合意形成する必要がある

テストを実行する前の段階をより詳細化してイメージできるようになろう

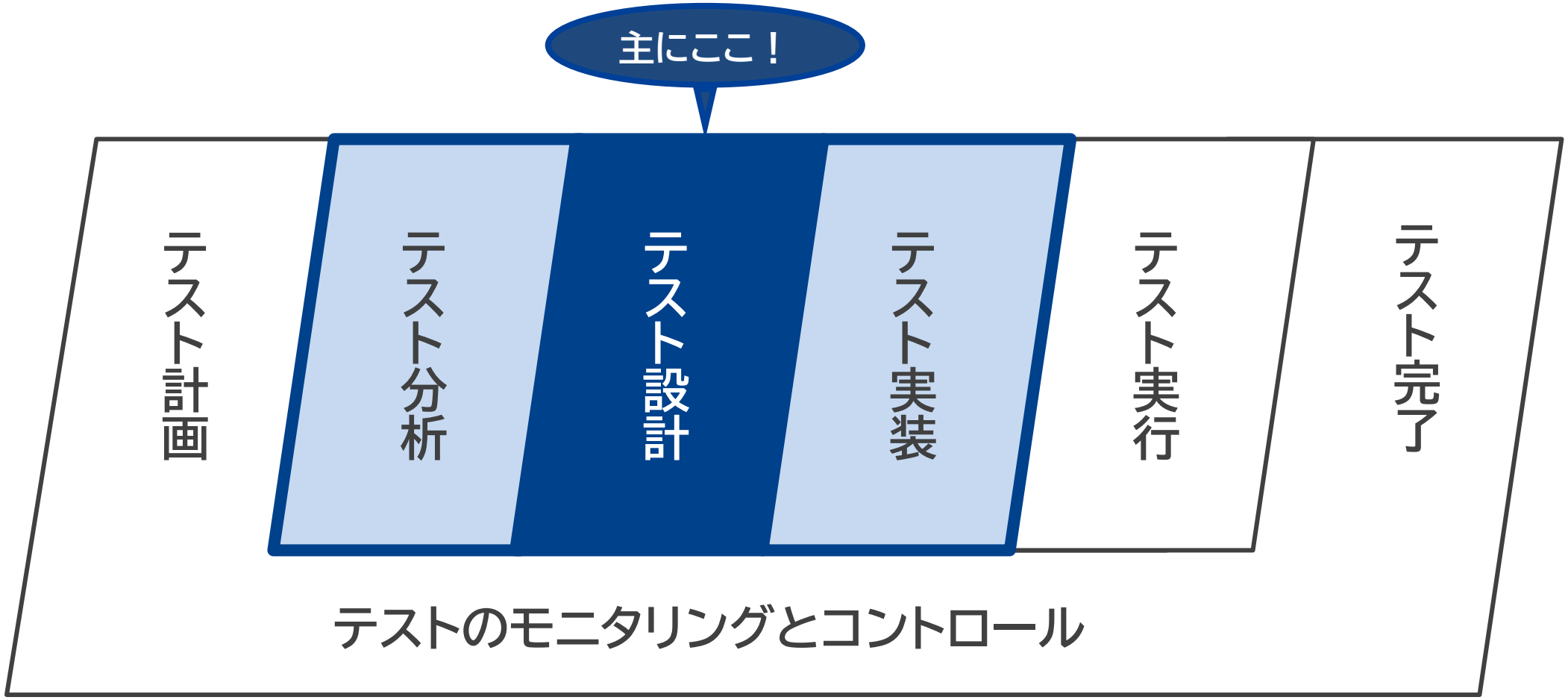


テストのライフサイクルにおける「テスト開発プロセス」の範囲



テスト開発 ＝ テスト分析 ＋ テスト設計 ＋ テスト実装

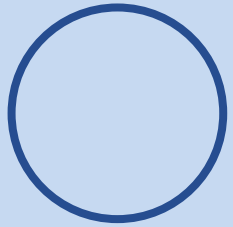
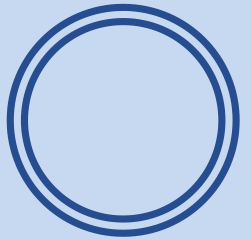
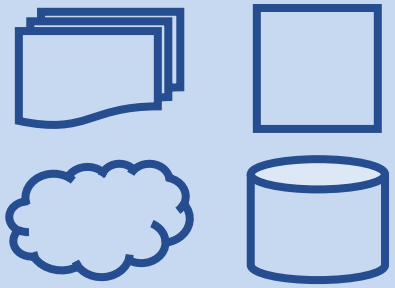
計画に基づいてテストを実行可能にするため、テストを開発する一連の活動を範囲としている。ただし、
現在のJSTQB FLシラバスでは「テスト開発」や「テスト開発プロセス」という用語は使われなくなっている点に留意。



テスト技法を使うのは主にテスト設計の段階であり、その前の
テスト分析が重要で、これを経ることでどの技法を使うか整理できる

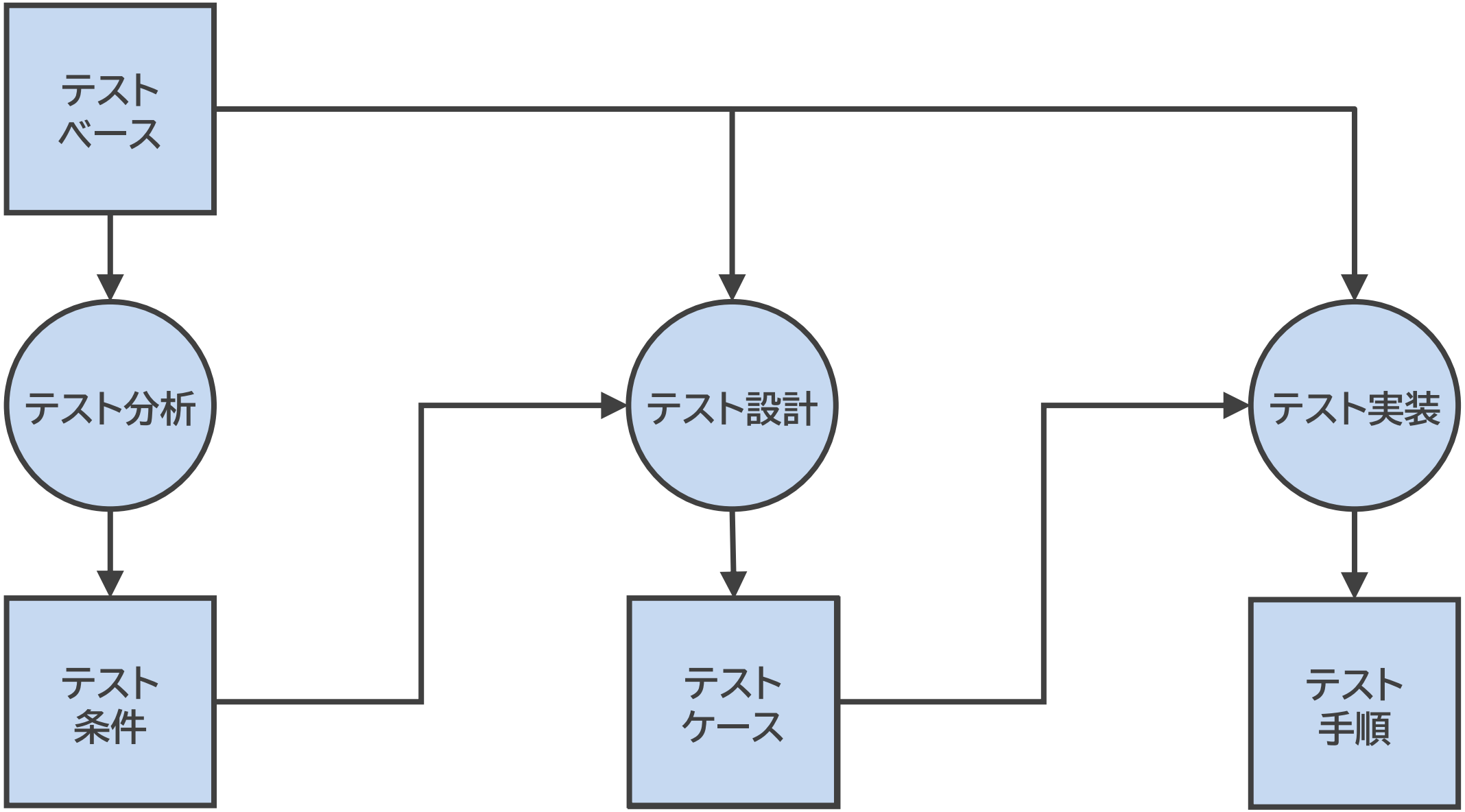
テスト開発プロセスを一番俯瞰したPFD

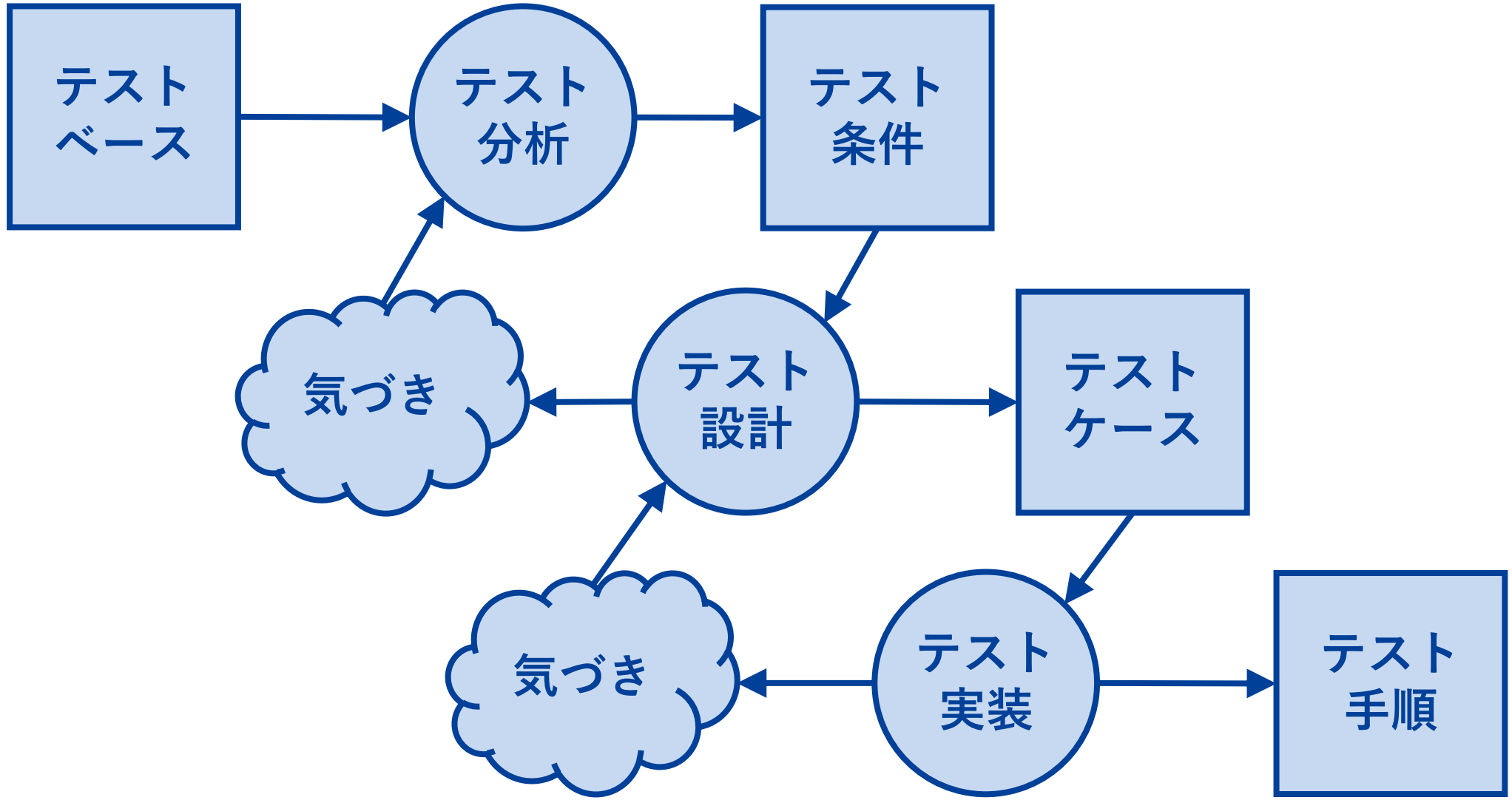
おおざっぱなPFD(Process Flow Diagram)の説明

要素	形状	説明	形状の別表現
プロセス		“作業を表します。 階層を持つ場合には線を太くしたり二重にすると良いでしょう	
成果物		“プロセスに出入りする成果物を表します。 ソースファイルや分冊の様子や、形になっていない「ノウハウ」など、適当な記号を使います。	
フロー		“成果物とプロセスを繋ぐ線。両側に矢印を付けて更新の意図をあわらしたり、線上に成果物を構成する要素を書くこともできます。	

出典: https://affordd.jp/koha hp/process/PFDform3.pdf P.16

テスト開発プロセスを深掘りしたPFD



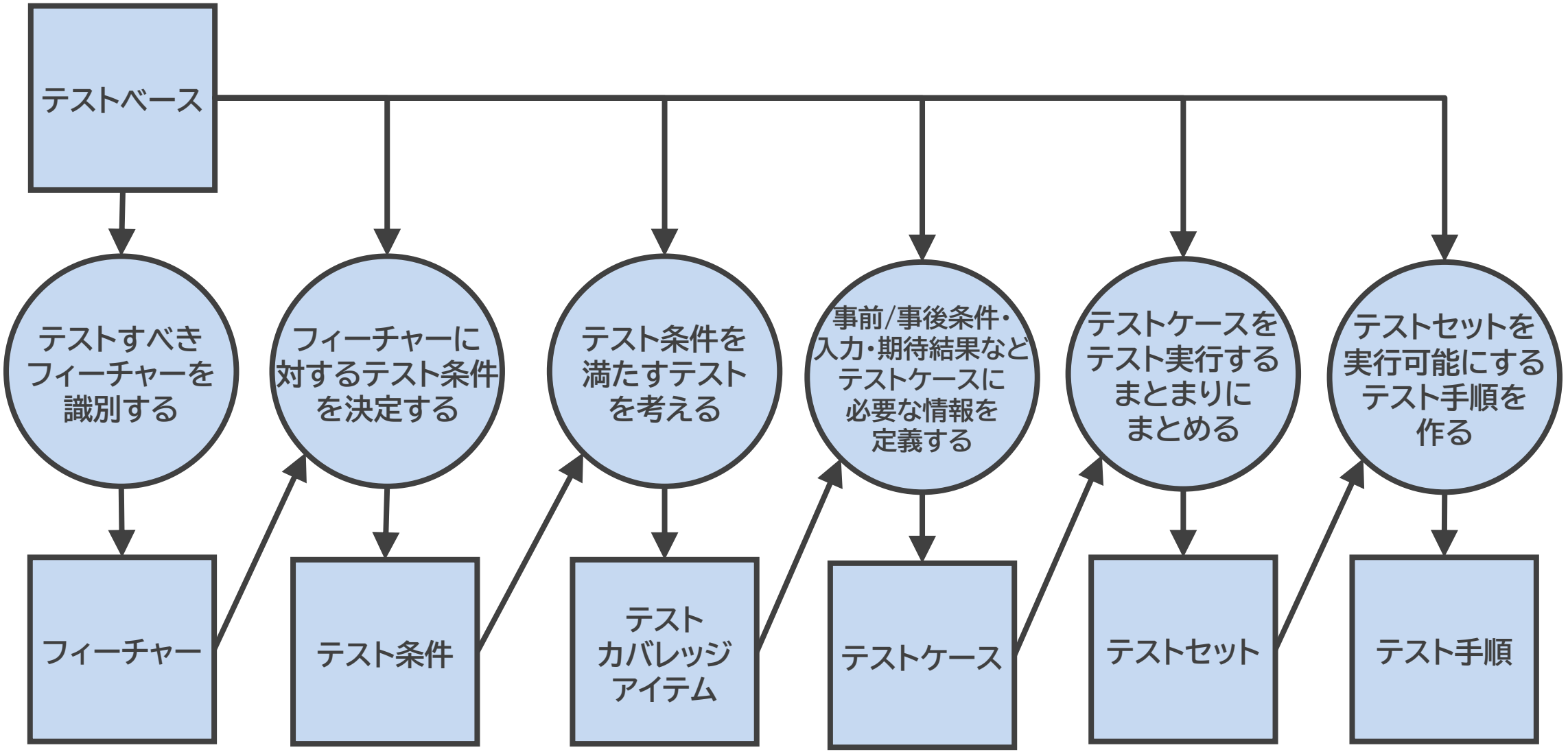


注意

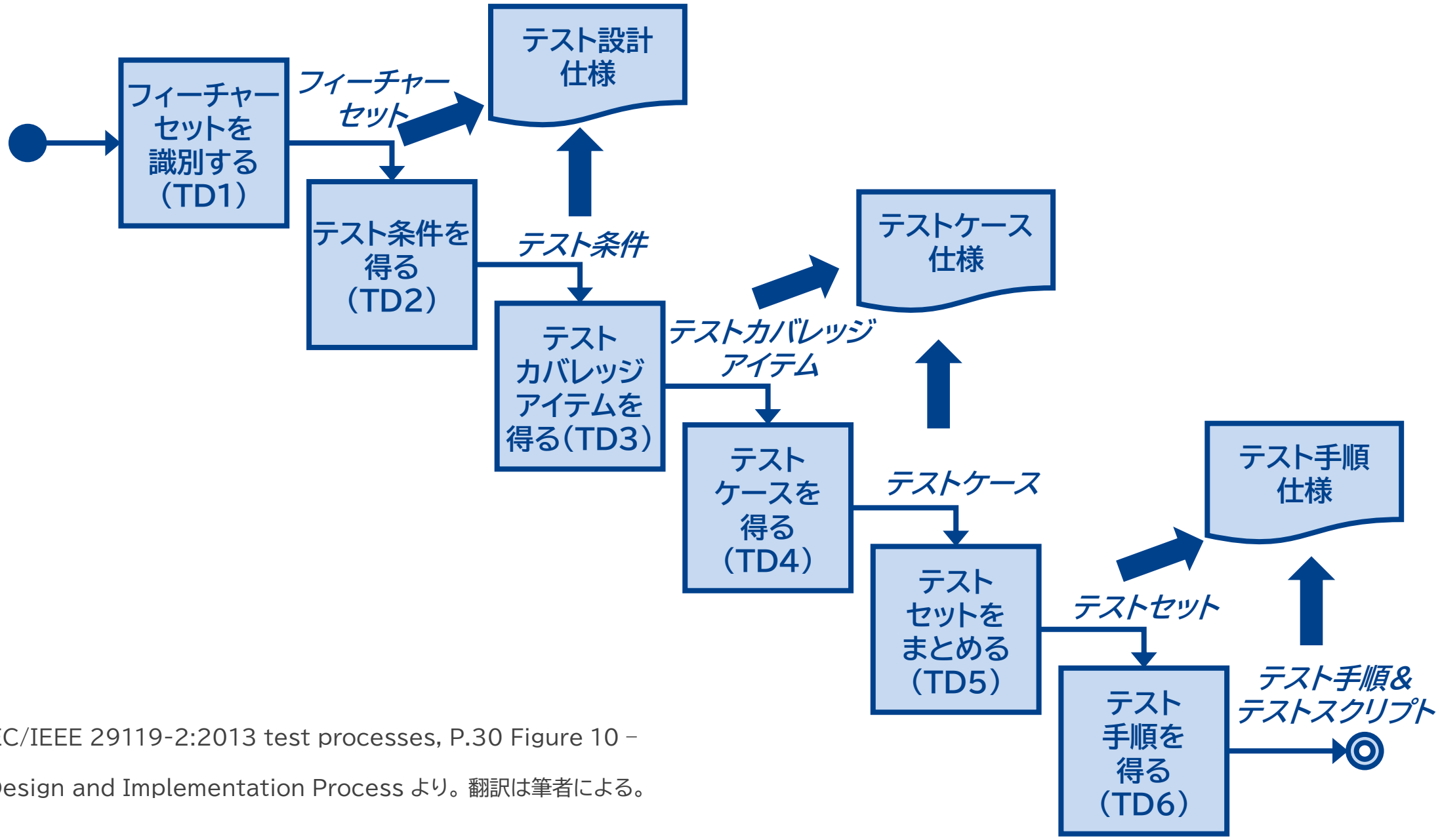
概念的にはテストプロセスとしての順序性があり、図にするとシーケンシャルに進めるように見えるが、実際には、分析→設計→実装をいったりきたりすることも多い

テスト開発プロセスをさらに深掘りしたPFD

※分かりやすくするためにテスト開発プロセスのすべてを表していない点にご留意ください

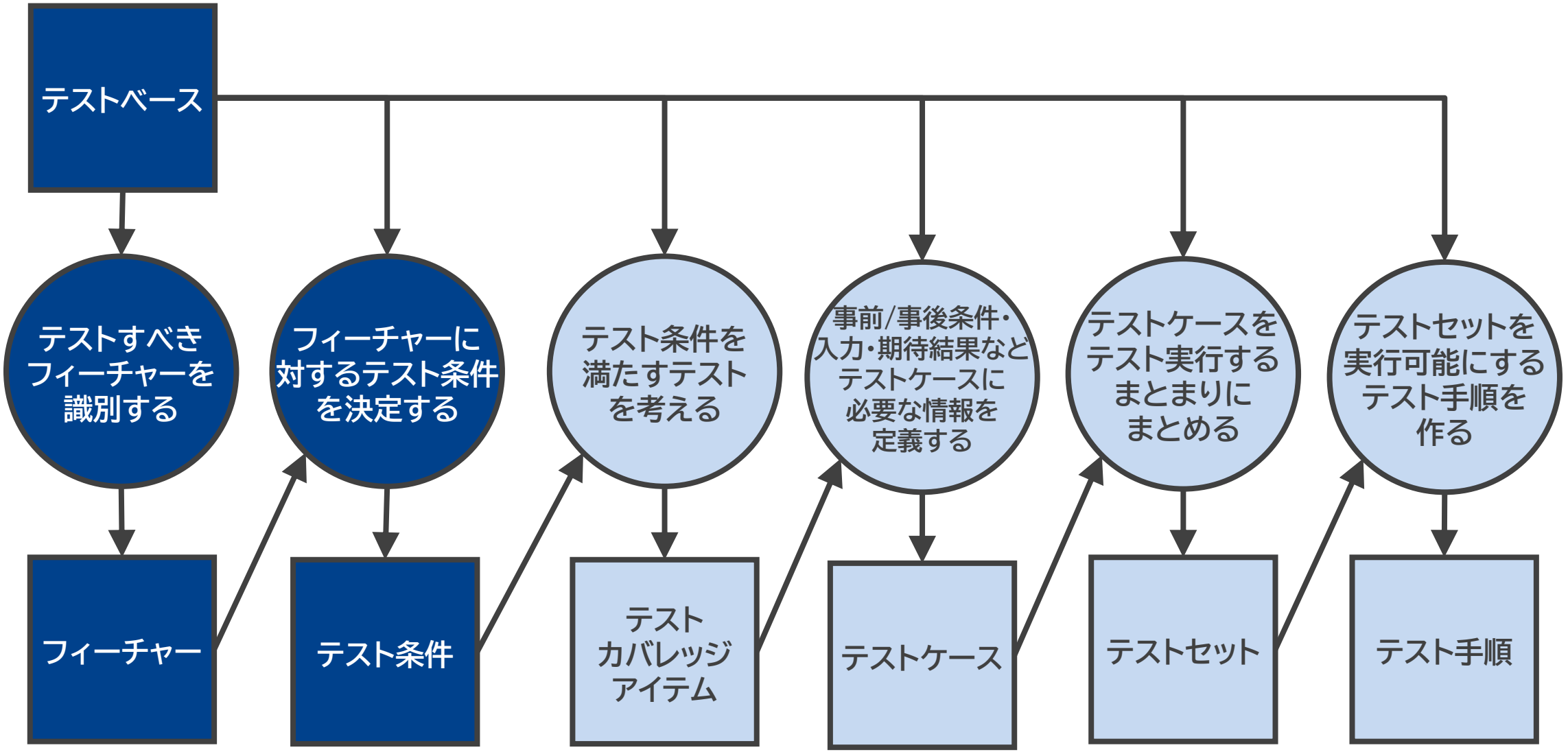


見え方は異なるが、ISO/IEC/IEEE 29119-2:2013のテスト設計と実装プロセスと基本的に同じ



ISO/IEC/IEEE 29119-2:2013 test processes, P.30 Figure 10 – Test Design and Implementation Process より。翻訳は筆者による。

テスト分析について、深掘りしてみる



とどのつまり、

フィーチャーと

テスト条件って

なんなの？

用語の定義を見ても、よく分からないですよね・・・・（定義も変わったりするし…）

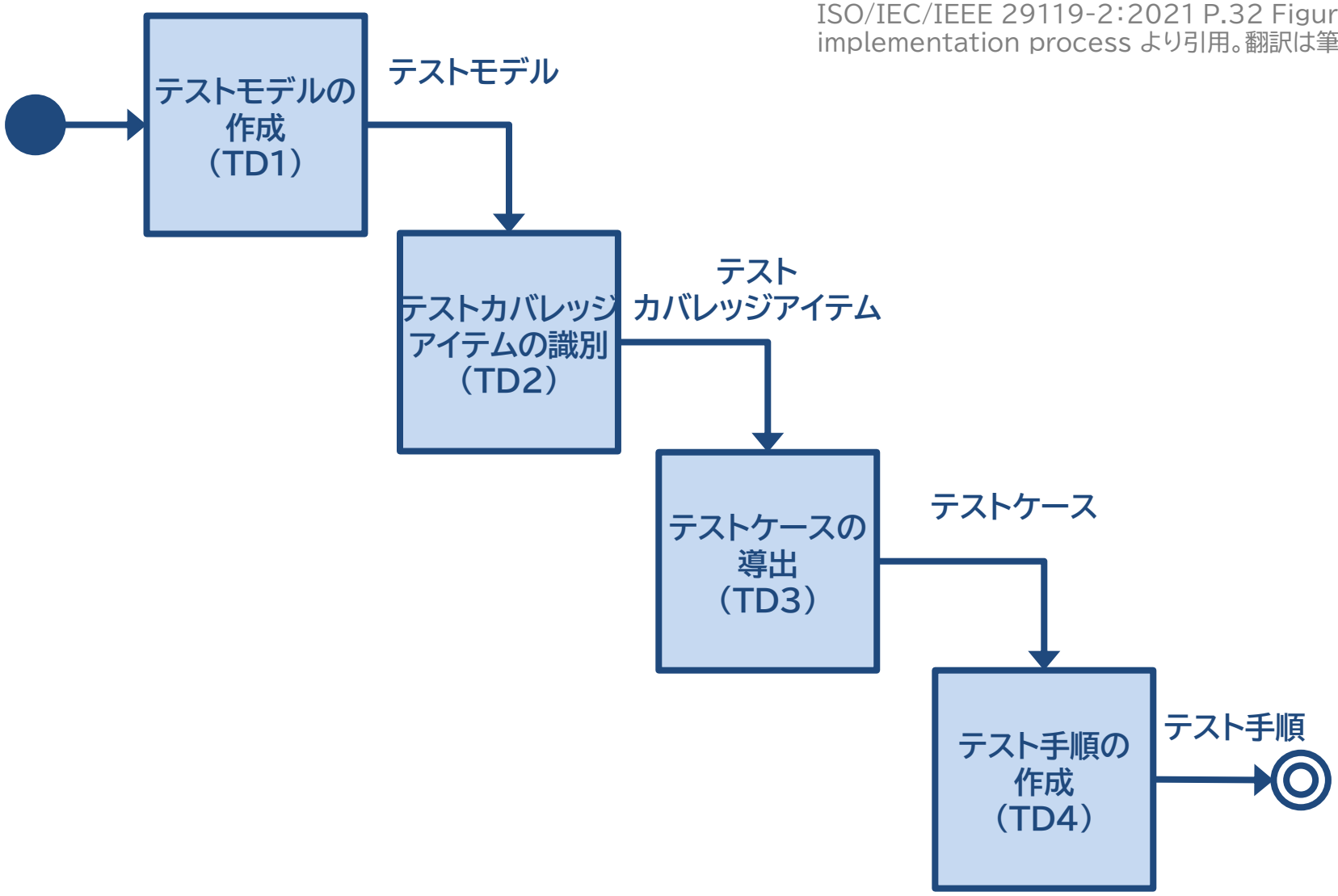
	JSTQBの用語定義	ISO/IEC/IEEE 29119での定義
フィーチャー	**フィーチャー：** 要求仕様ドキュメントで、明示的、暗示的に規定したコンポーネントやシステムの属性（たとえば、信頼性、使用性、設計上の制約など）。 ※最新のISTQBの用語集では、featureの定義がなくなっている	**フィーチャーセット：** 後続のテスト設計活動でほかのフィーチャーセットとは独立して扱うことのできる、論理的なテストアイテムの部分集合。
テスト条件	• コンポーネントやシステムのアイテムやイベントで、テストケースにより検証できるもの。たとえば、機能、トランザクション、フィーチャ、品質特性、構造要素など（**昔の定義**） • テストのための根拠となる、コンポーネントやシステムのテストが可能な一部分（**新しい定義**）	コンポーネントやシステムにおけるテストの根拠として識別された、機能、トランザクション、フィーチャー、品質特性、構造要素など、テスト可能な側面。

誤解を恐れずに言えば、フィーチャーやテスト条件に関する分かり易い統一的な見解はない

最新のISO/IEC/IEEE 29119-2:2021の「テスト設計と実装プロセス」では

テスト条件じゃなくテストモデルに変わっちゃいましたしね…

ISO/IEC/IEEE 29119-2:2021 P.32 Figure 10 Test design and implementation process より引用。翻訳は筆者による



この記事は最終更新日から1年以上が経過しています。

ベリサーブ Advent Calendar 2021 21日目

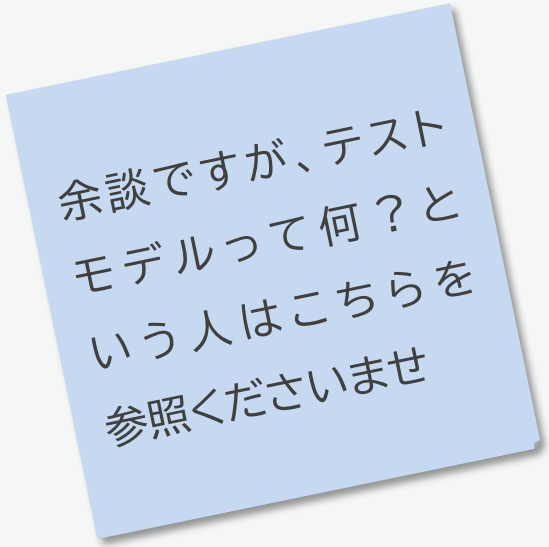
@yamasaki696 (山﨑 崇) in 株式会社ベリサーブ
投稿日 2021年12月21日　更新日 2021年12月22日　2570 views

ISO/IEC/IEEE 29119-2～4の改訂で新しくでてきたテストモデルについて少し調べてみた

ソフトウェアテスト, 国際規格, 29119

3 Lines Summary

- 2021年10月に、ISO/IEC/IEEE 29119 Part 2～4の改訂版がリリースされました
- その中で、テストモデルという新しい概念が示されたようです
- テストモデルってなんだろうかを調べてみたのでそれを整理してみました



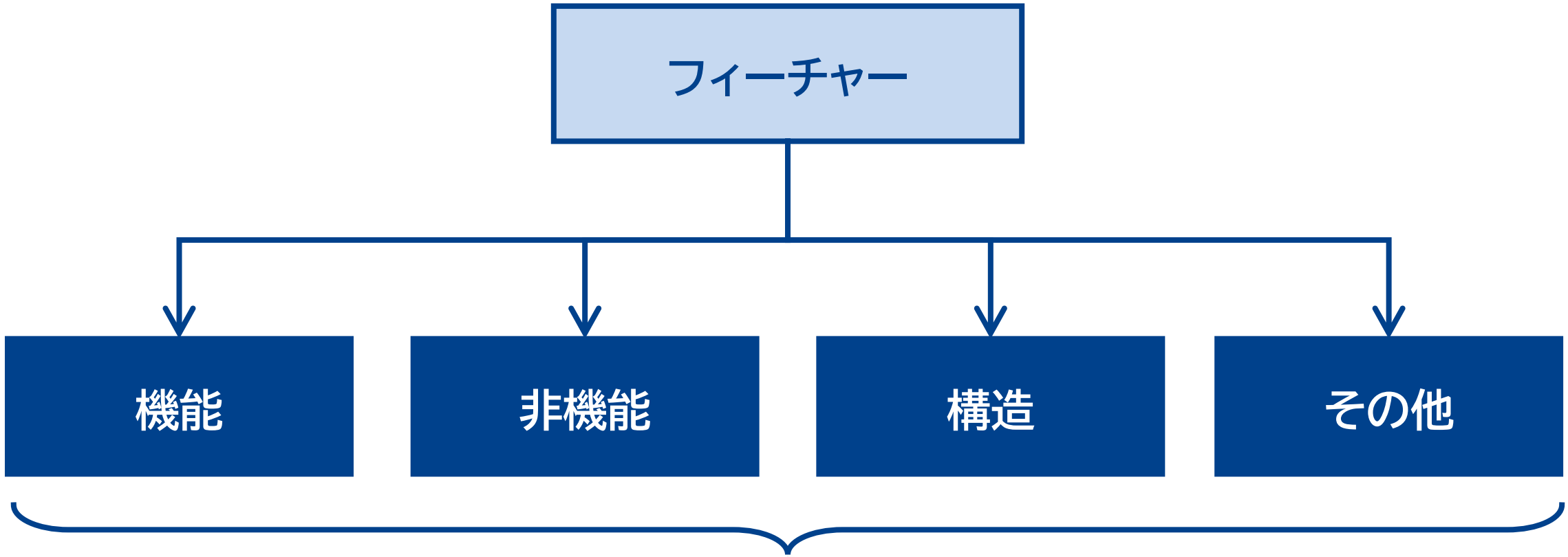
発想を転換しよう！

JSTQBやISO/IEC/IEEE 29119の定義に沿っているかどうかではなく、フィーチャーに何を置いて、テスト条件に何を置くと、自分の考えたテストを説明しやすくなり、利害関係者と合意形成しやすくなるのかで決めてみよう

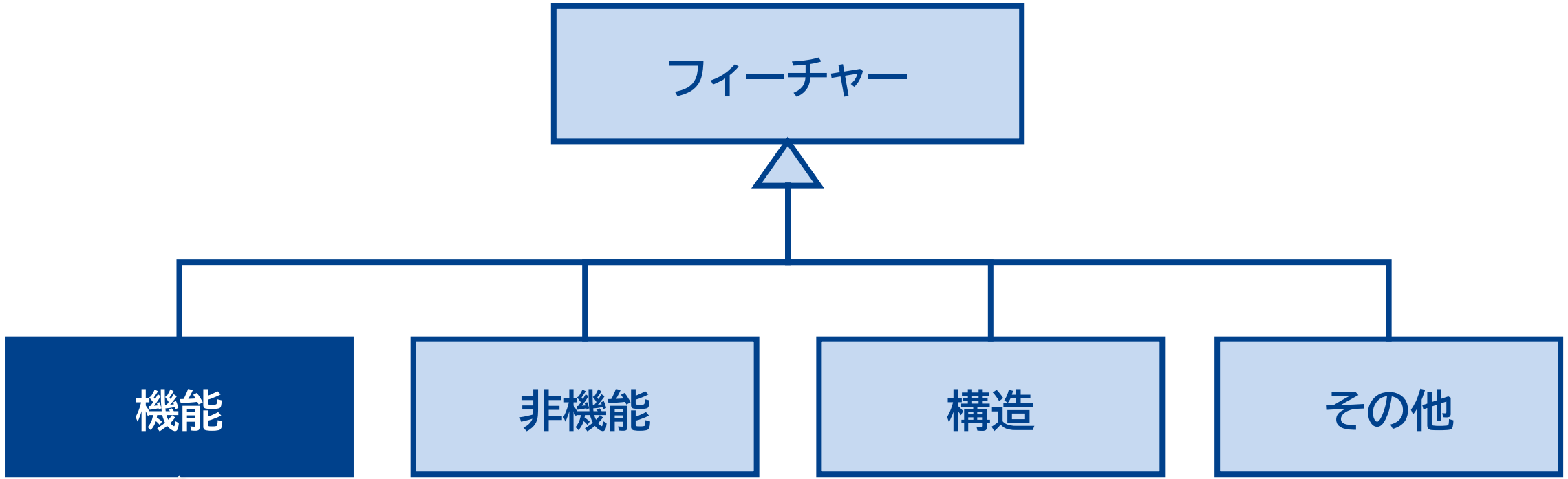
フィーチャー

フィーチャー（feature）と**厳密にで対応する日本語はない**。

あえて言えば「特徴」や「特性」であり、テスト対象が持つ特徴であれば、**何でもフィーチャーとして捉えることができる**

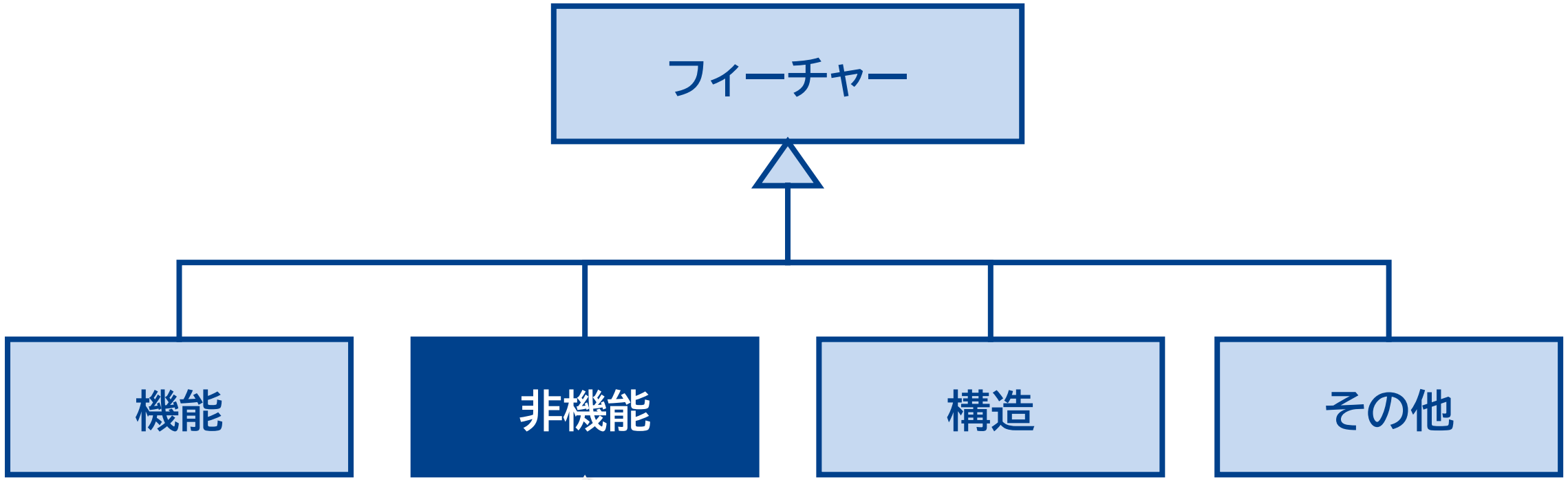


誤解を恐れずに例を示すと、フィーチャーは**機能**や**非機能**や**構造**
（もちろん、**それだけに限らない**）であるとしてみる



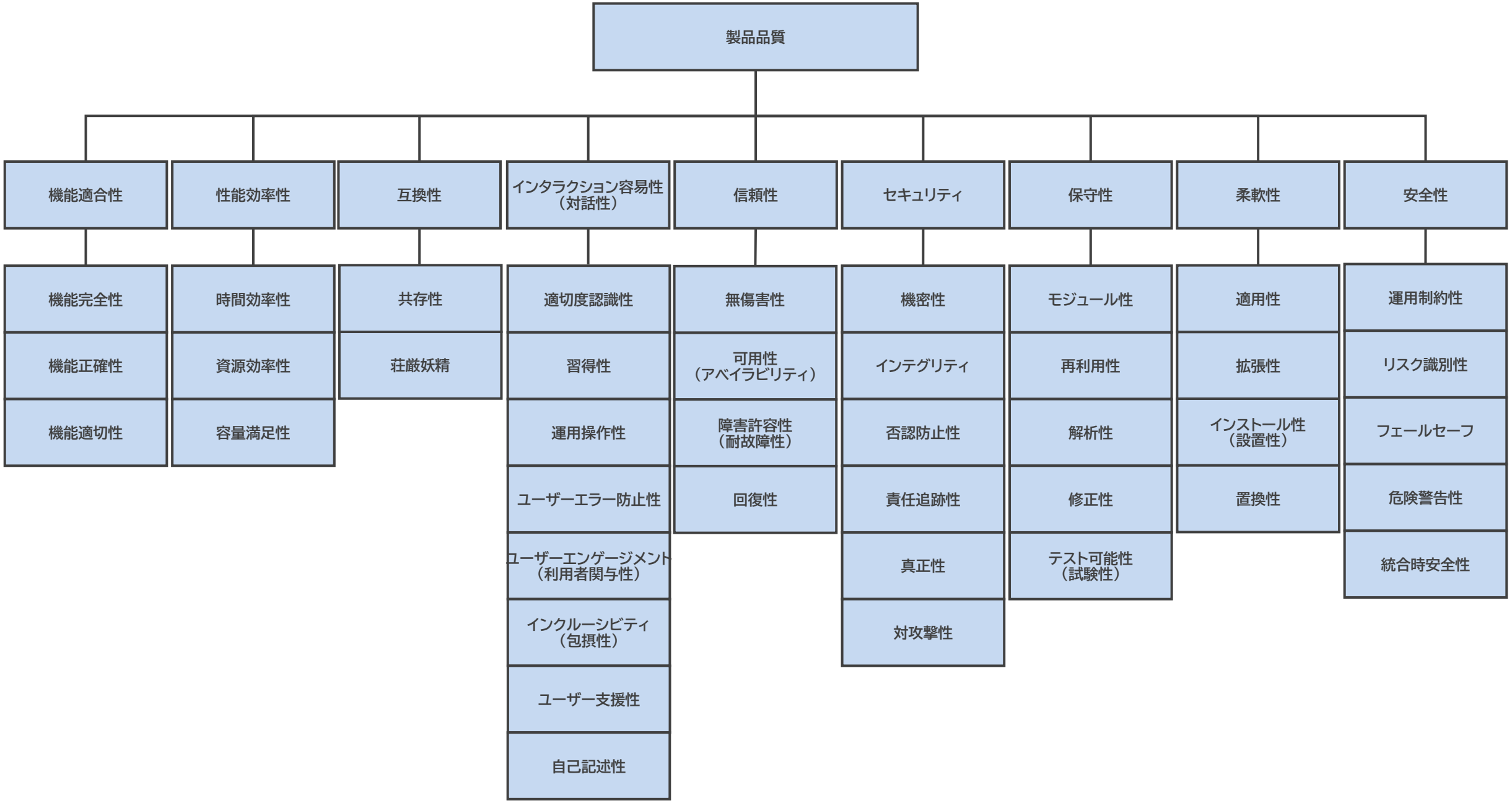
システムテストの**機能テスト**でテスト分析するなら、例えば**フィーチャーセット**は**システムが備える機能の一覧**であって良い

もちろん、テストベースで〇〇機能と記述されるものが、システムテストの機能テストにおいて扱う機能の単位になっているかは限らないので、**テストの視点から機能を再整理**したほうが良い場合もある

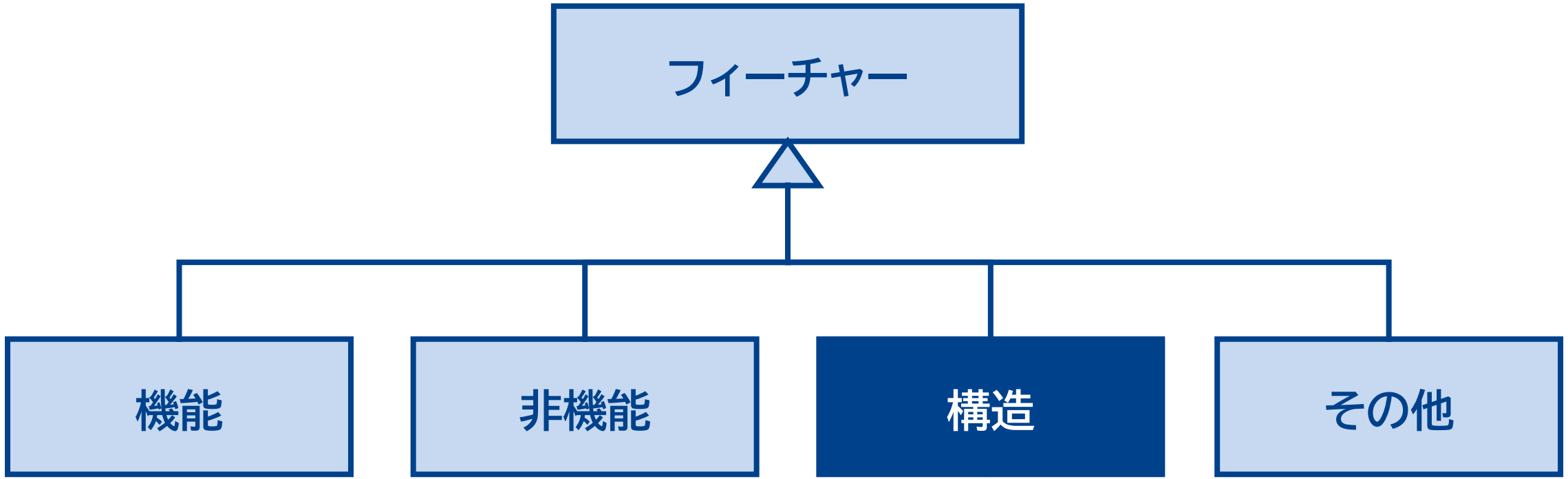


非機能テストで、テスト技法を使わないわけではもちろんないが、非機能テストでは技法以上に、テストプラクティスが重要になることが多い。

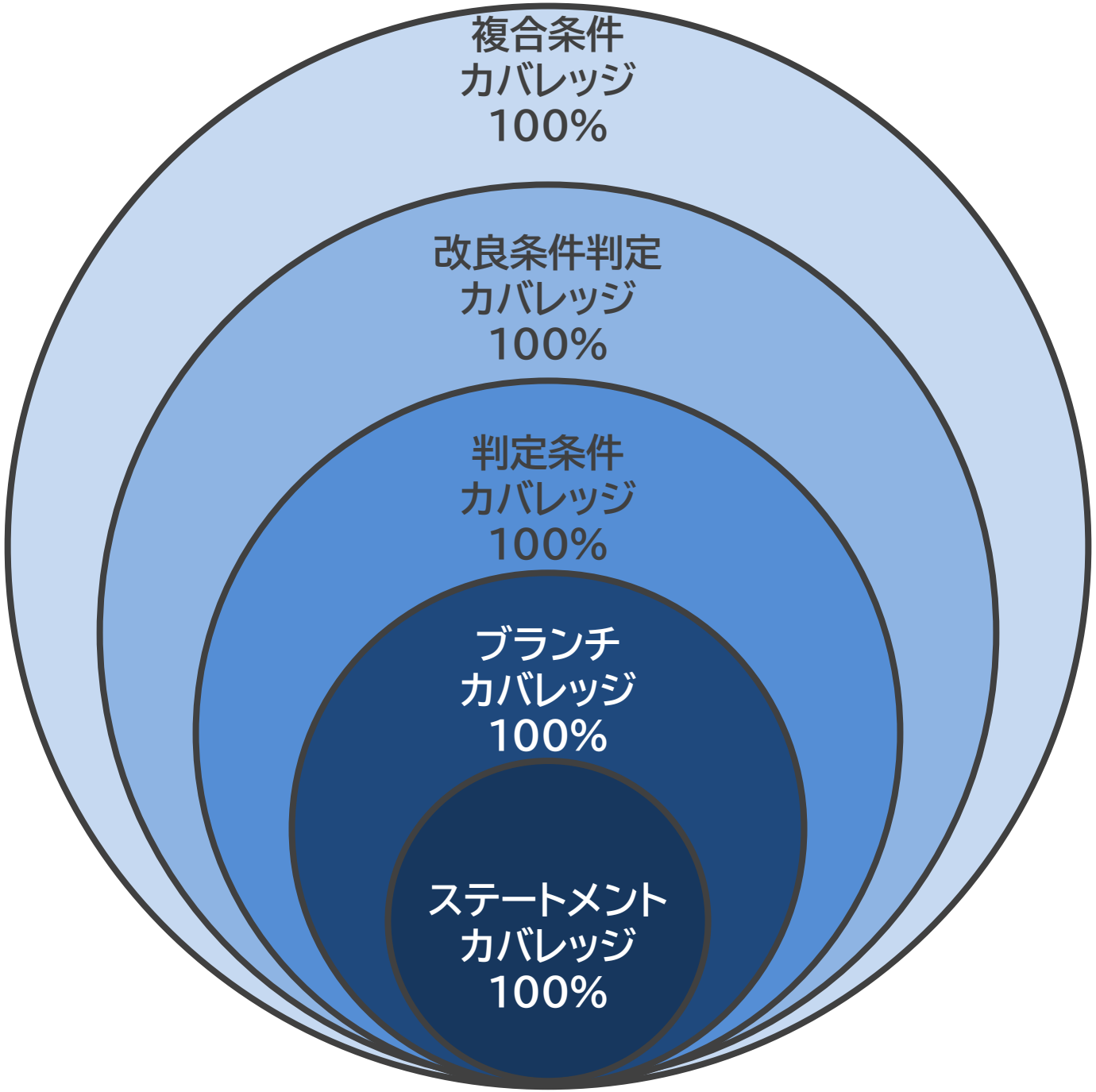
例えば、セキュリティテストであれば、ペネトレーションテストやファズテスト、
性能テストであれば、性能の特性（負荷、ストレス、拡張性、スパイク、耐久など）に応じた環境とツールの構築など。

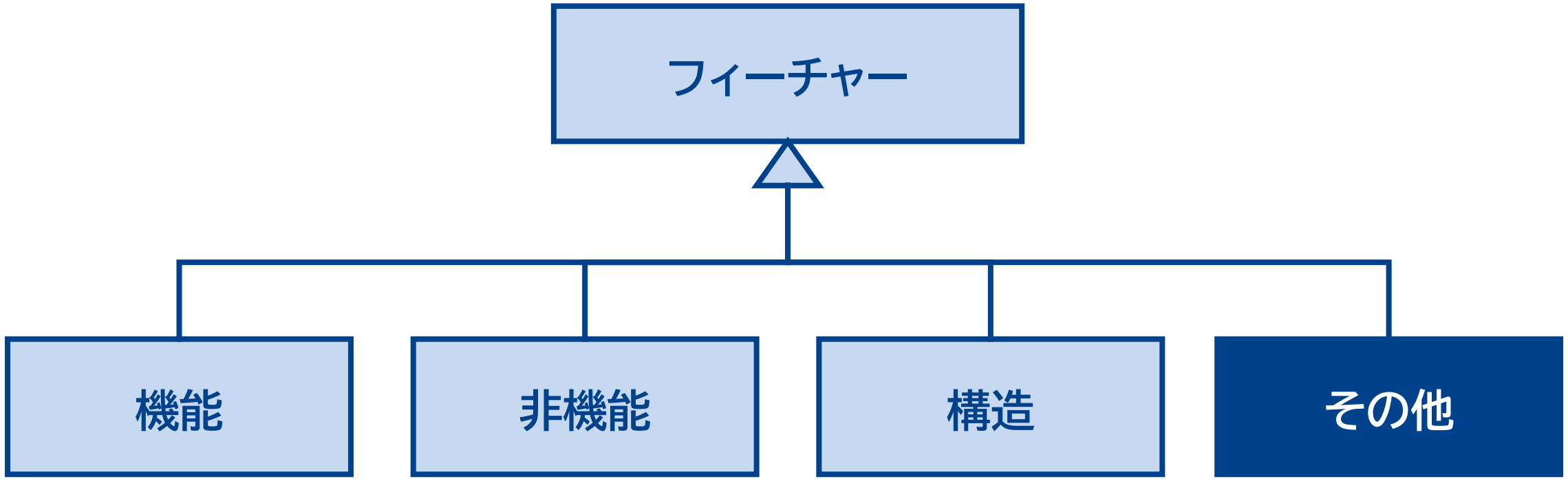


出典; JIS X 25010:2025の情報を元に、独自に作図



構造などは一番わかりやすく、ソースコードの構造であれば、どのコードカバレッジの基準を選ぶのか（ステートメント、ブランチ、判定条件、MC/DC、複合条件など）、決めの問題。

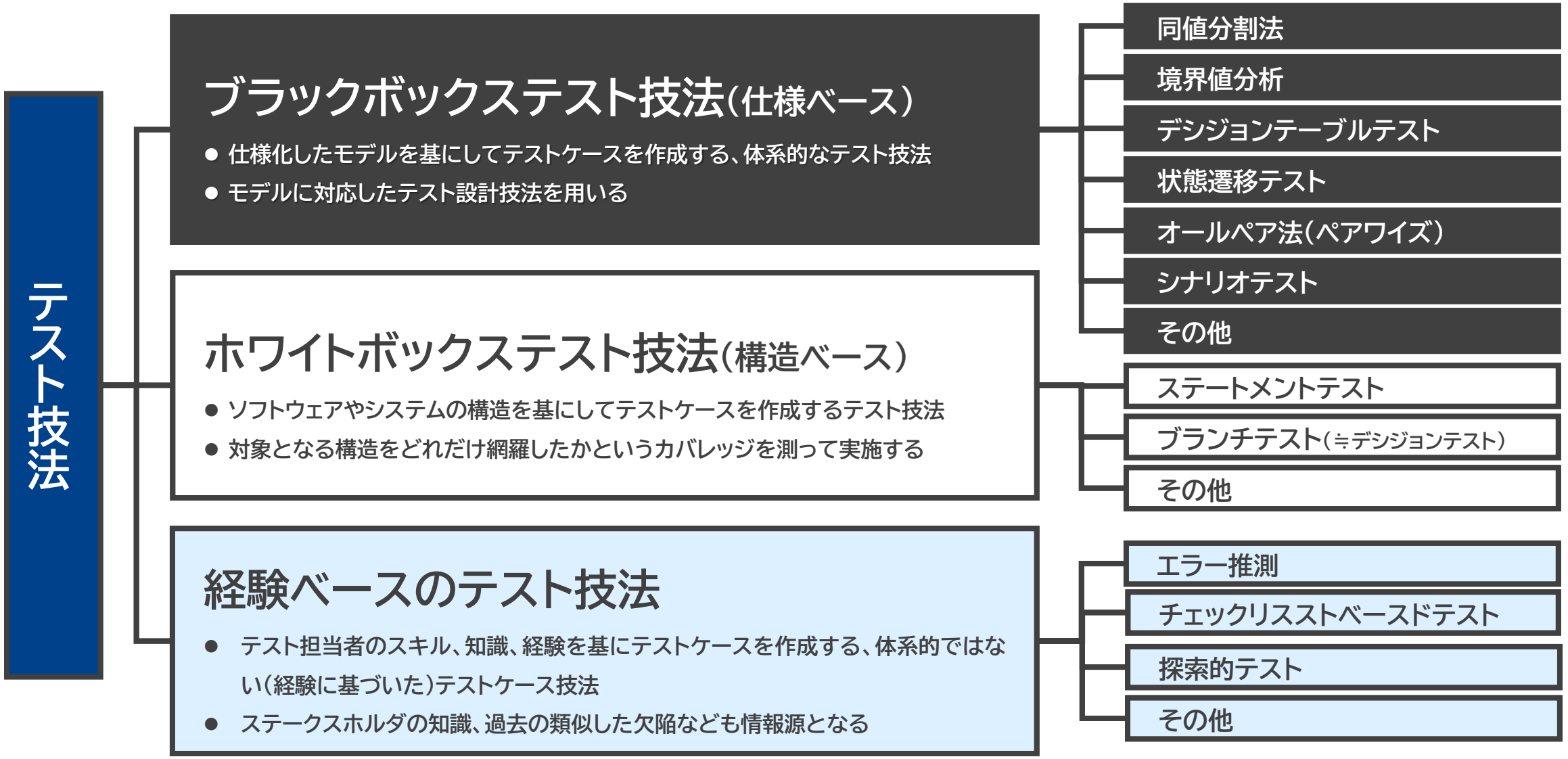

複合条件
カバレッジ
100%
改良条件判定
カバレッジ
100%
判定条件
カバレッジ
100%
ブランチ
カバレッジ
100%
ステートメント
カバレッジ
100%

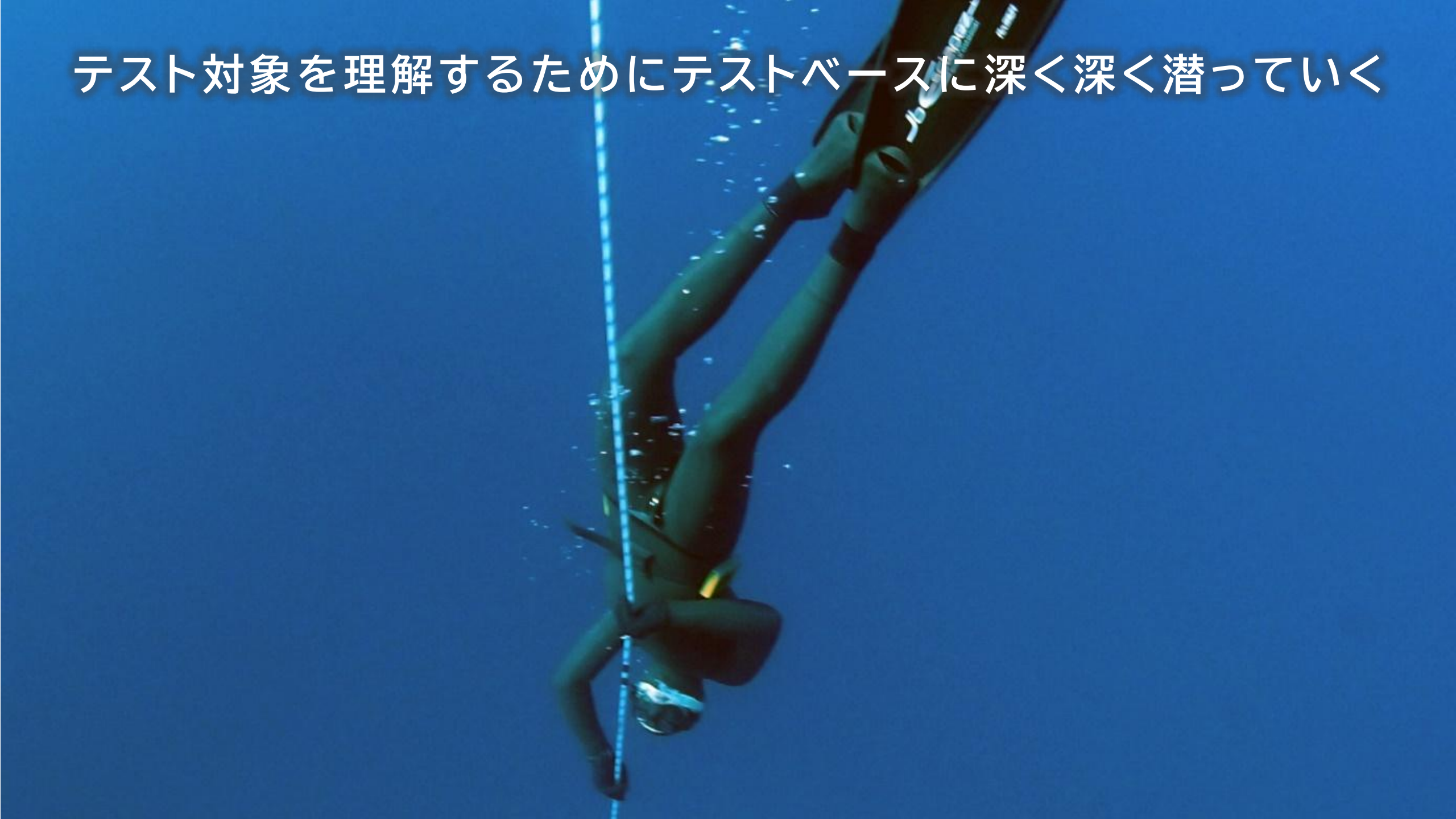


必ずしも、機能、非機能、構造に限らなくても良い。

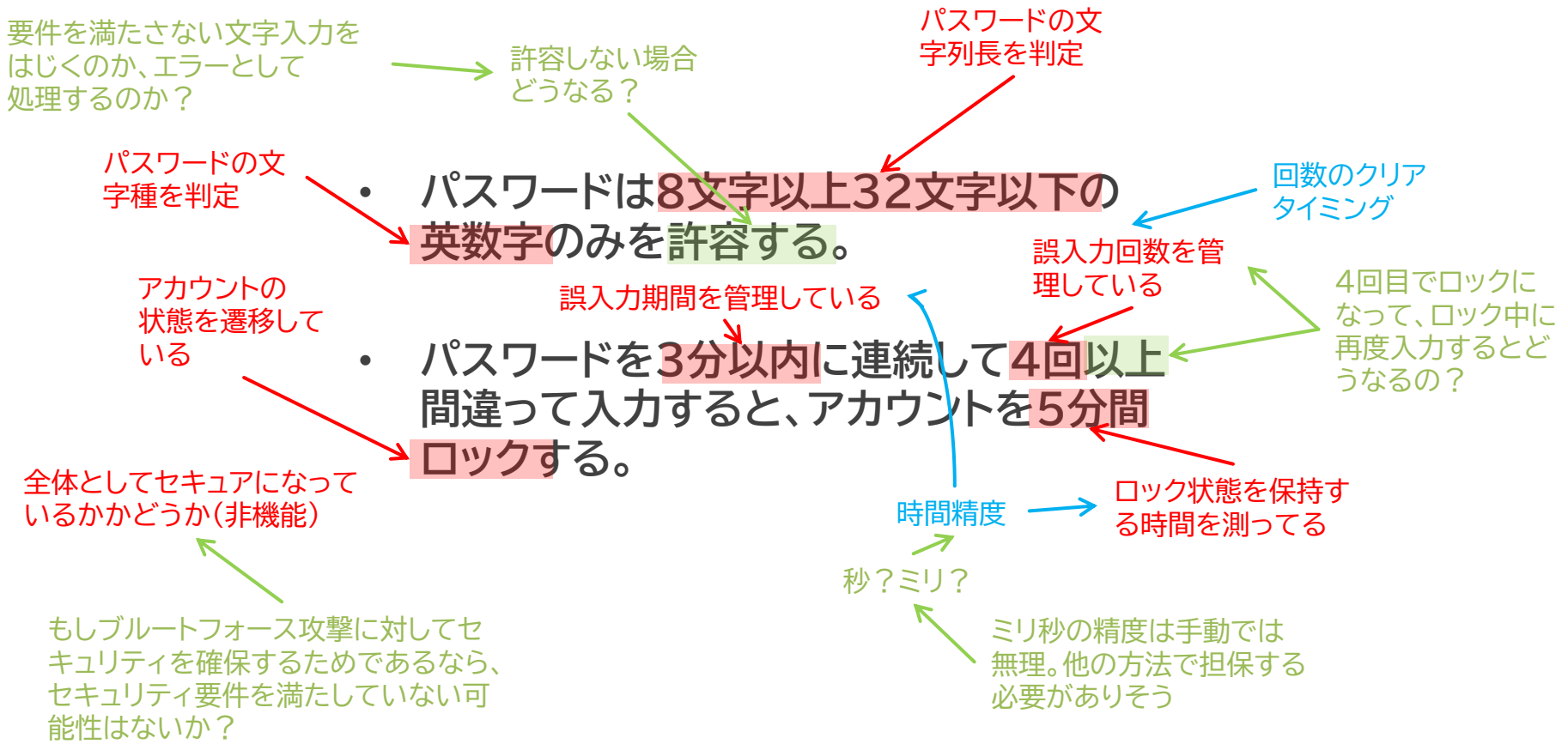
例えば、網羅的な観点以外にも、ピンポイント的なテストも検討することも多いし、過去のトラブル事例などにもとづき経験ベースのテスト技法を用いたり、ガイドワードなどを使って起きて欲しくないことなどをあぶり出してもよい。

技法の特徴を抑えておくことで、どう言ったところでどの技法を使うと良いかも把握できる




テスト対象を理解するためにテストベースに深く深く潜っていく

3色ボールペンでテストベースの情報を分析・補完した例



どれだけテストベースを汚せるかがポイントで、不明部分をテストの視点から洗い出す。
この時点で曖昧性や矛盾などを取り除くとともに書かれていないことについても深堀する。

3色ボールペンの使い方の例

出典:「ソフトウェア・テストPRESS Vol.2」三色ボールペンで読む仕様書(鈴木三紀夫氏)

赤

- 客観的に見て、最も重要な箇所
- たとえば、フィーチャーやテスト条件

など

青

- 客観的に見て、まあ重要な箇所
- たとえば、フィーチャーやテスト条件を構成する要素

など

緑

- 主観的に見て、おかしいと感じた箇所
- たとえば不明点、疑問点

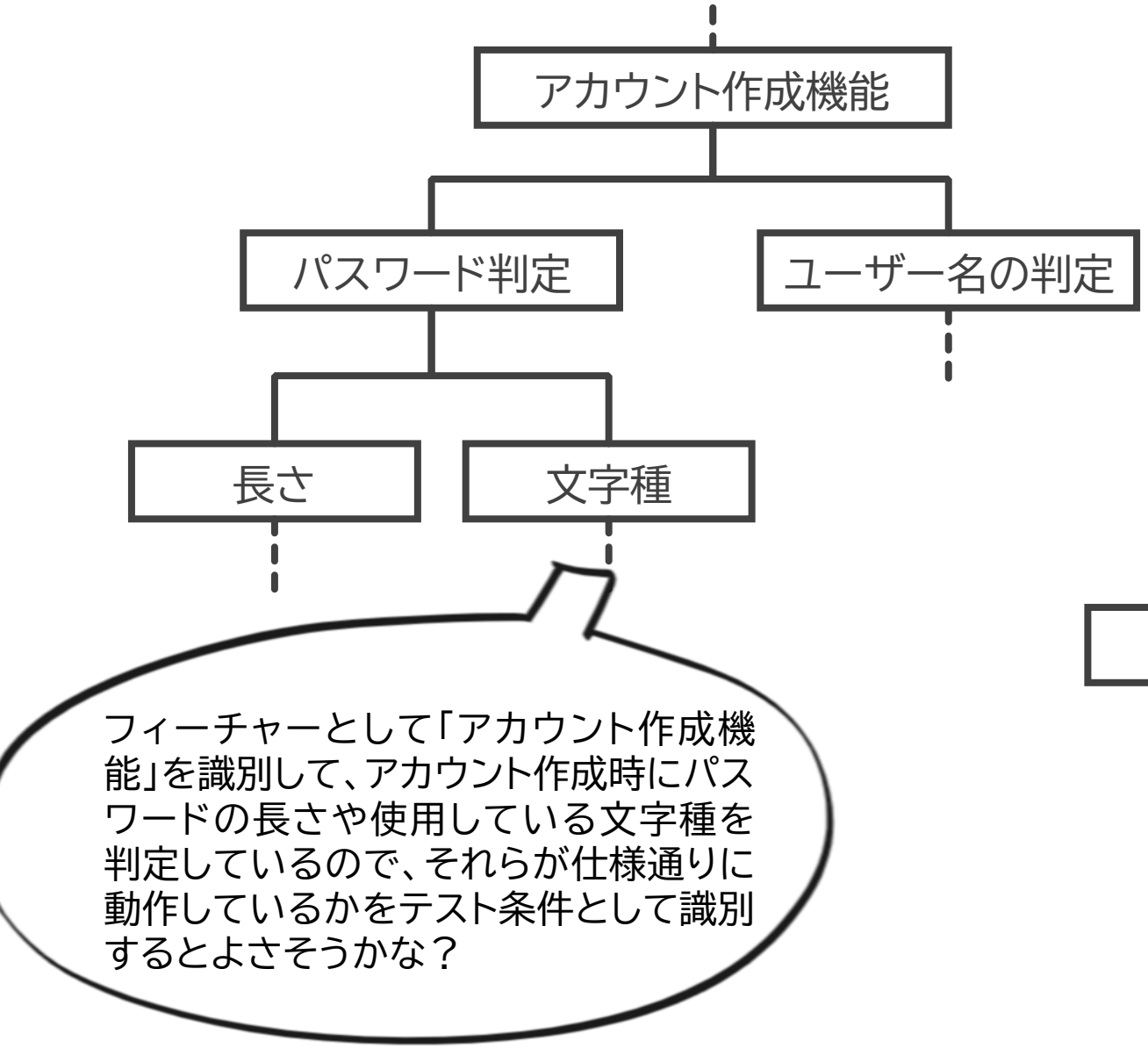
など

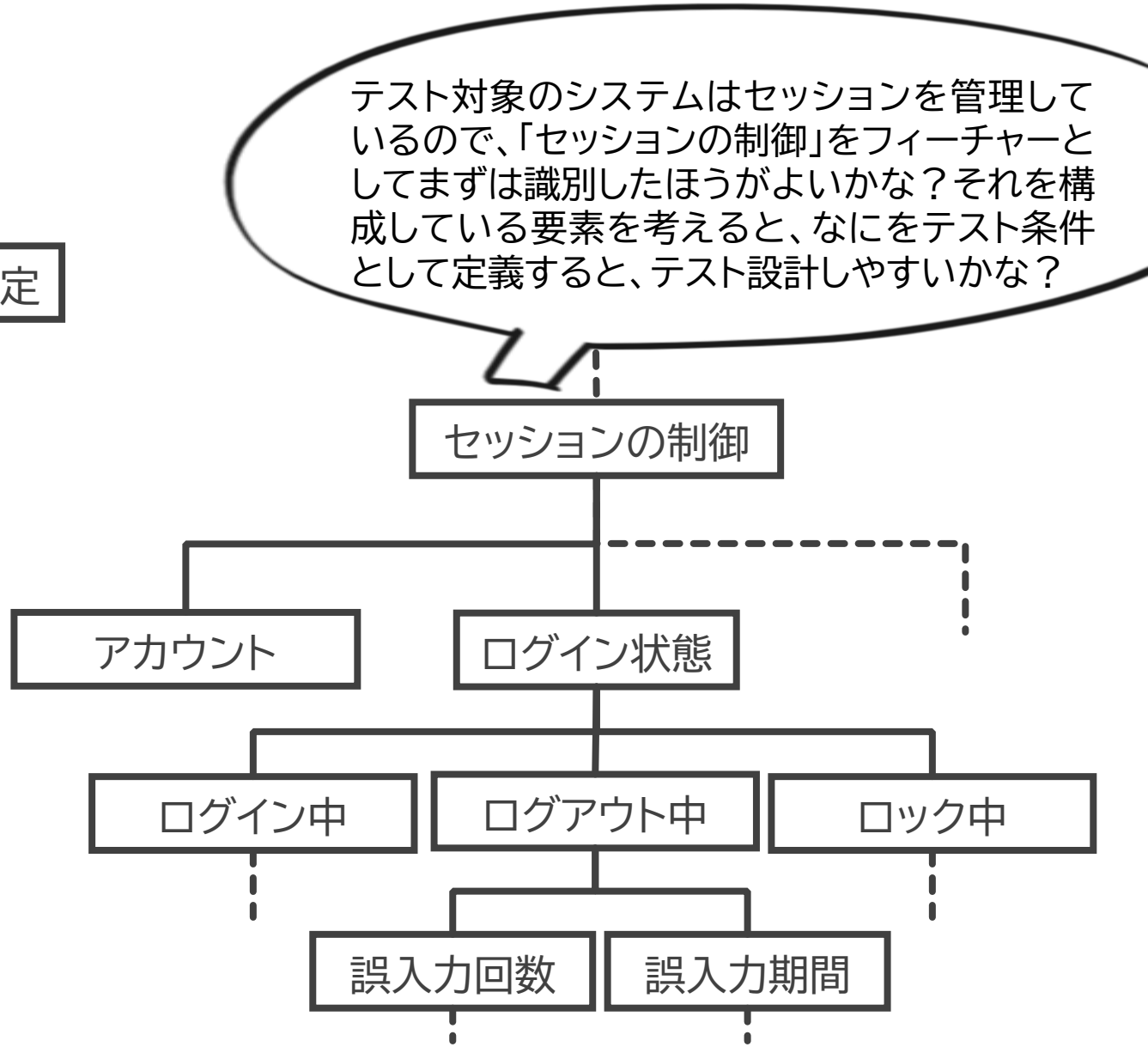
正しく色を使い分けることを意識しすぎると、仕様書を汚せなくなるのであくまで色は目安。
色の選択に悩んでしまうくらいであれば、単色でもよいので仕様書を汚すことに注力しよう。

なぜ仕様書を汚して、情報を分析するのか？

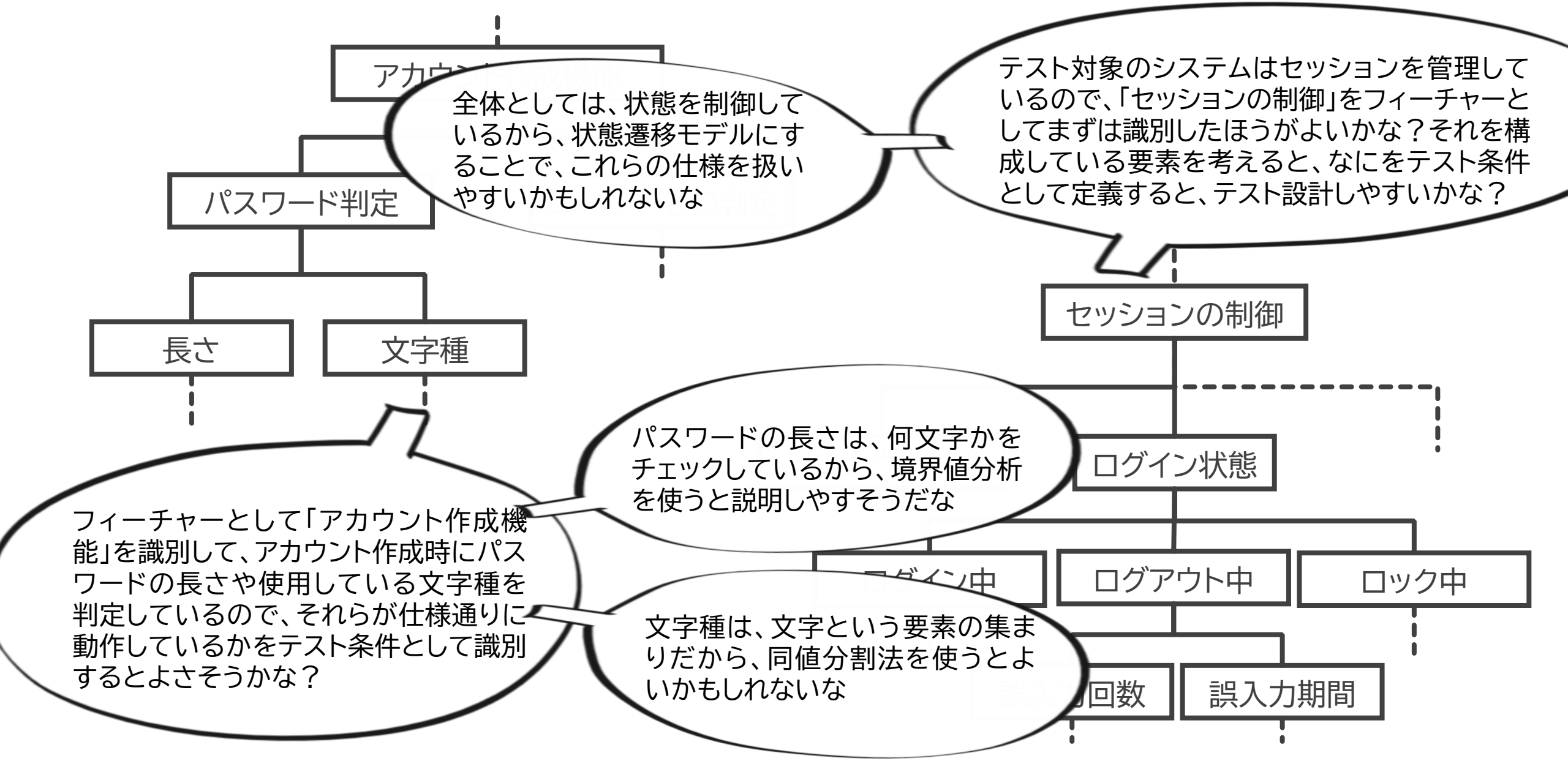
- 多くの場合、分析対象のテストベースの仕様書などは十全ではない
  - テストに必要なすべての情報が記述されているわけではない
  - 抜け漏れや誤りを含んでいることもある
- 多くの場合、分析対象のテストベースはテストのために作られていない
  - 仕様書や設計書は第一義に作るのための文書であり、開発とテストでは関心事が異なる
  - 開発のために構造化された情報は、テストを考えて実行する上での構造と異なることも多い

3色ボールペンで汚した仕様書を基に再整理した例

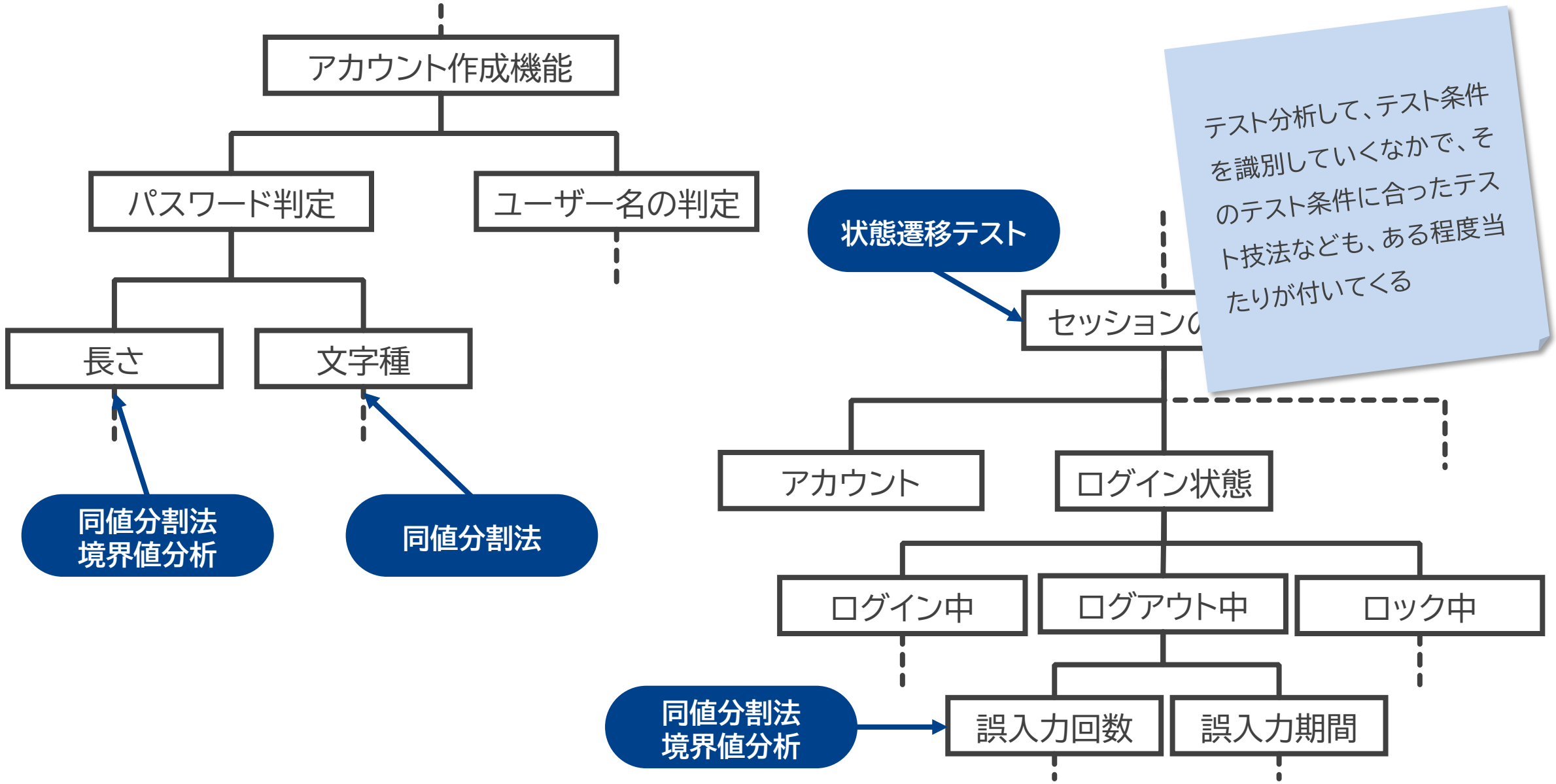




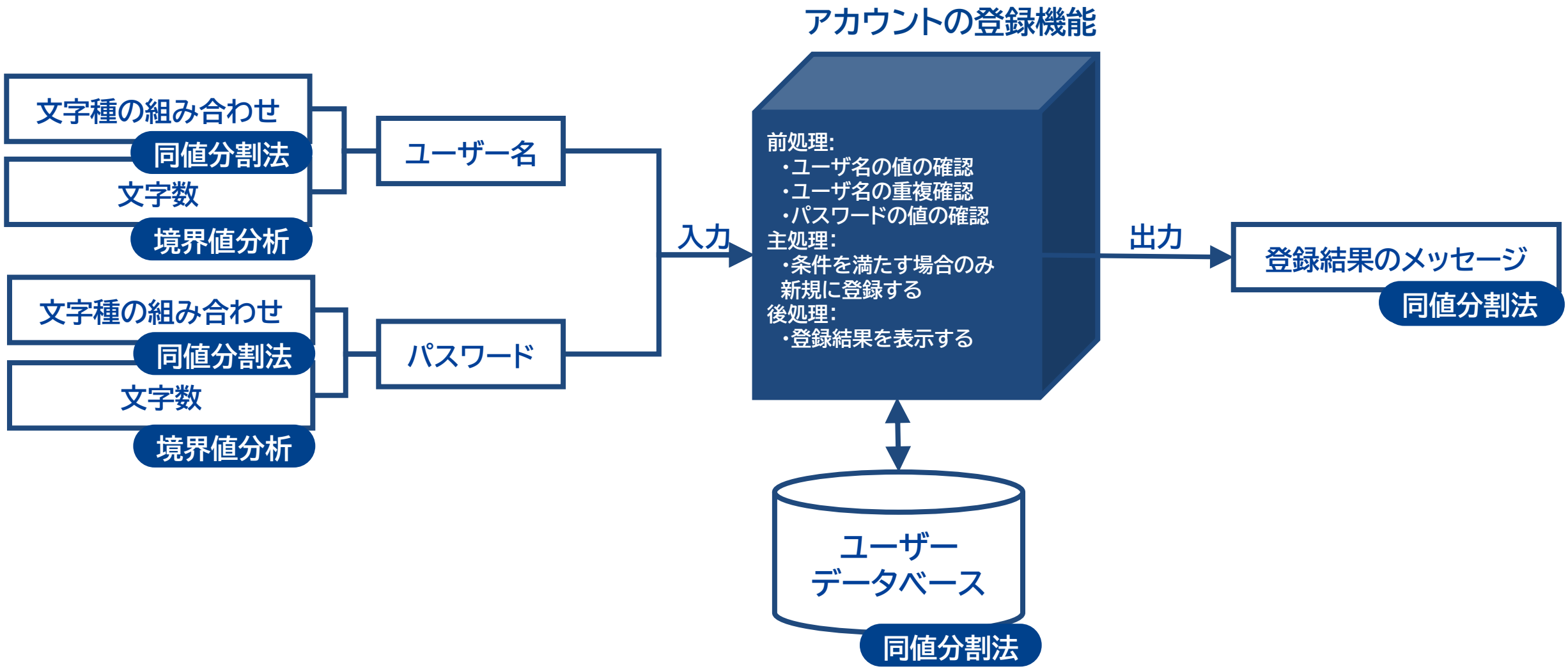
3色ボールペンで汚した仕様書を基に構造化した例



整理していく中で、テスト技法を適用できる箇所が識別できる

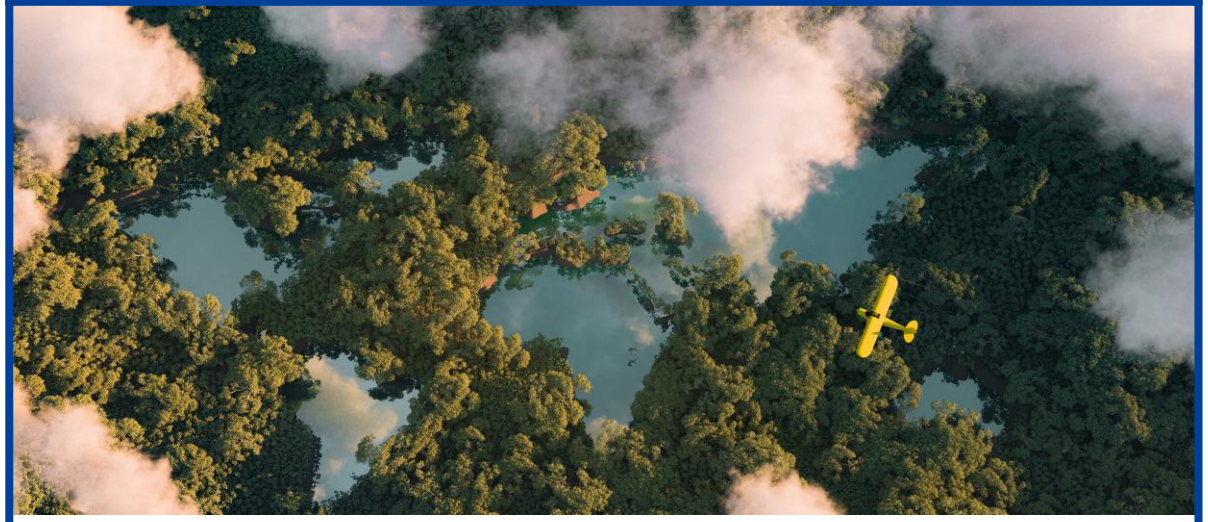


機能とその入出力関係を整理した例



なぜ構造化するのか？

全体を俯瞰しやすい



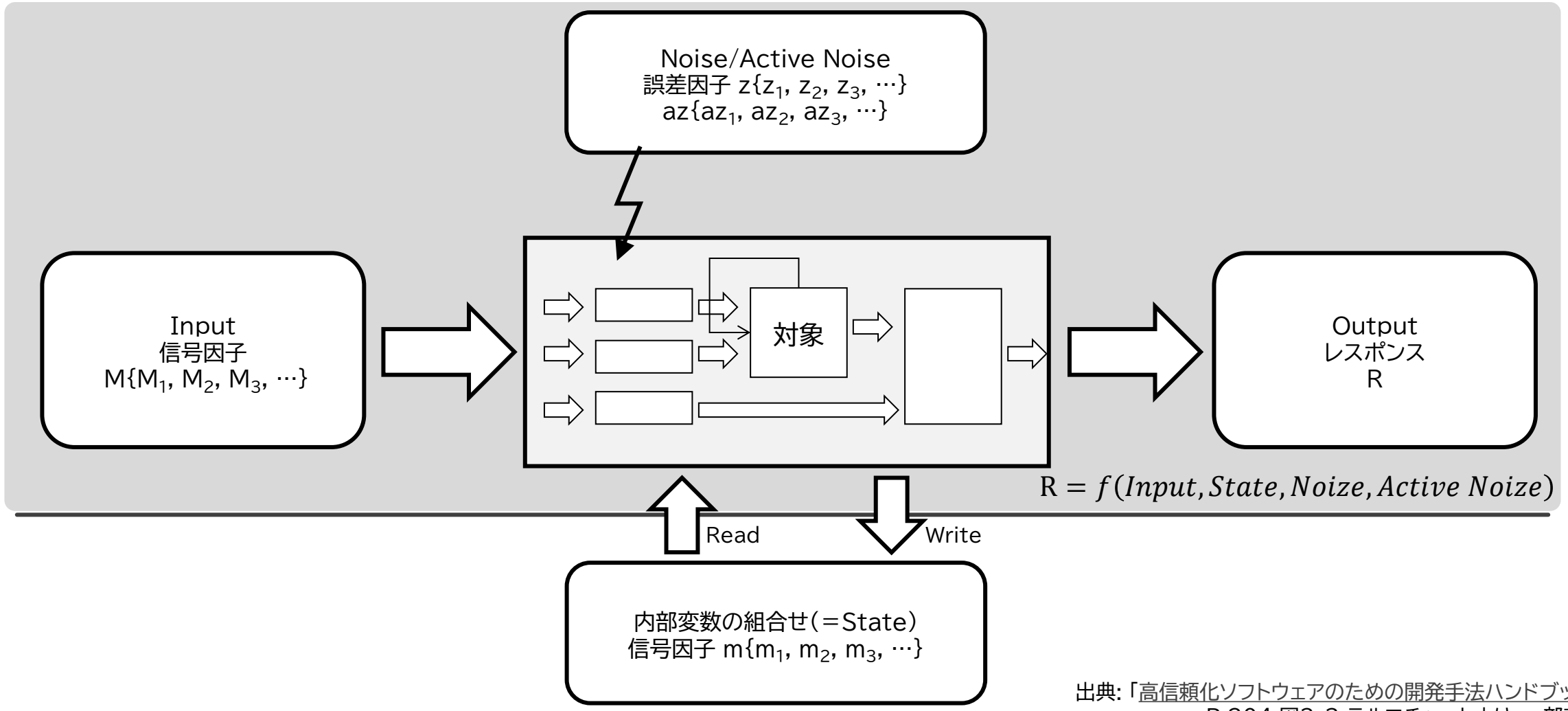
- 大量のテストケースやテスト手順は枝葉であり全体を俯瞰し難い
- 木や森を見るには関心事に応じた抽象度のほうが分かりやすい
- そして、ステークホルダ毎に関心事は異なる

構造化した結果に基づいて議論しやすい

- ぱっと見て把握しやすい
- 関係性が分かりやすい
- MECE（漏れなくダブりなく）であるかを把握しやすい

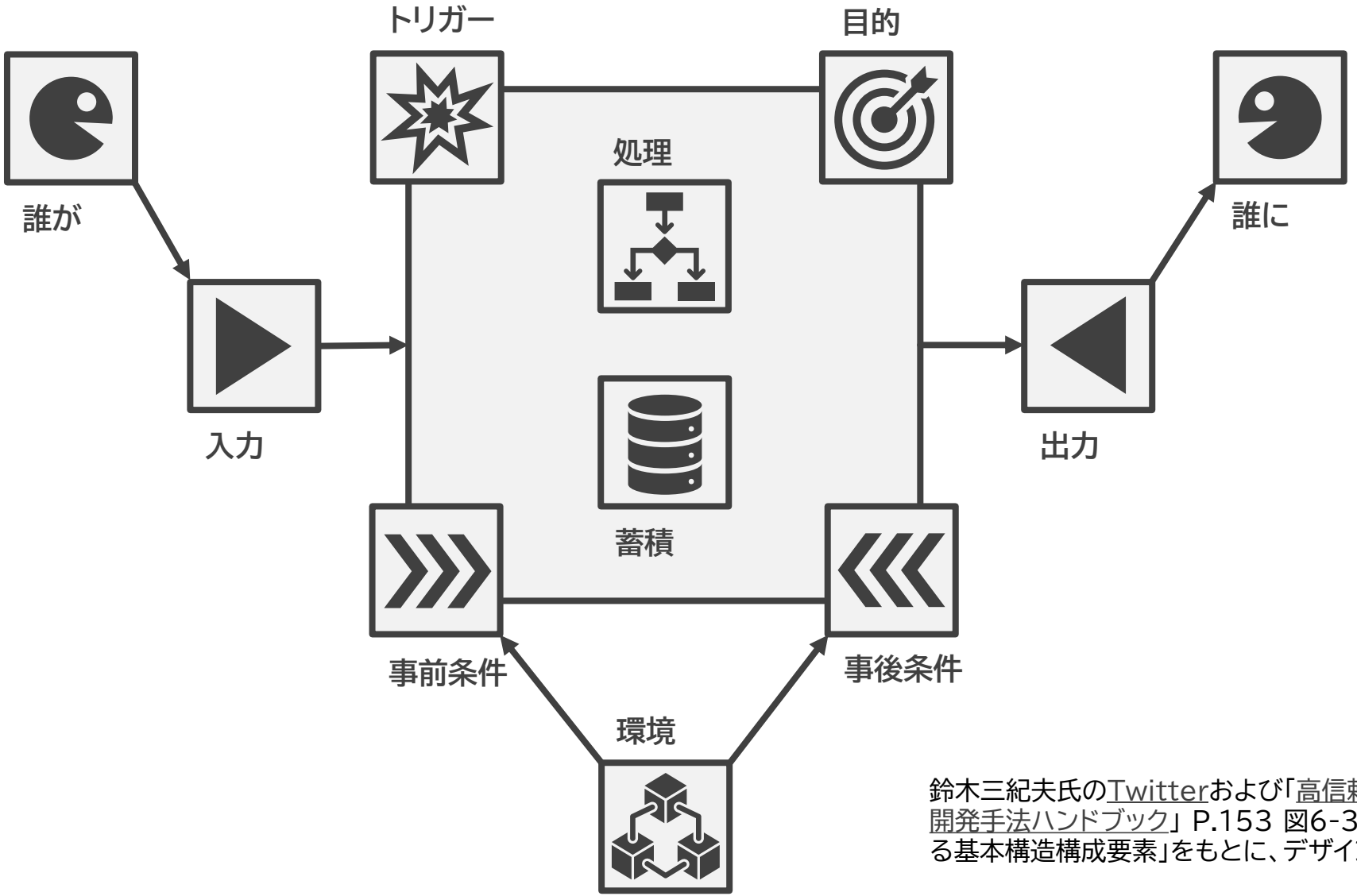
など

HAYST法によるラルフチャート



出典:「高信頼化ソフトウェアのための開発手法ハンドブック」
P.204 図3-3 ラルフチャートより。一部変更

Tiramisによる基本構造構成要素



鈴木三紀夫氏のTwitterおよび「高信頼化ソフトウェアのための開発手法ハンドブック」P.153 図6-3「Tiramisで使用している基本構造構成要素」をもとに、デザインを変更して掲載

ゆもつよメソッドによる論理的機能構造

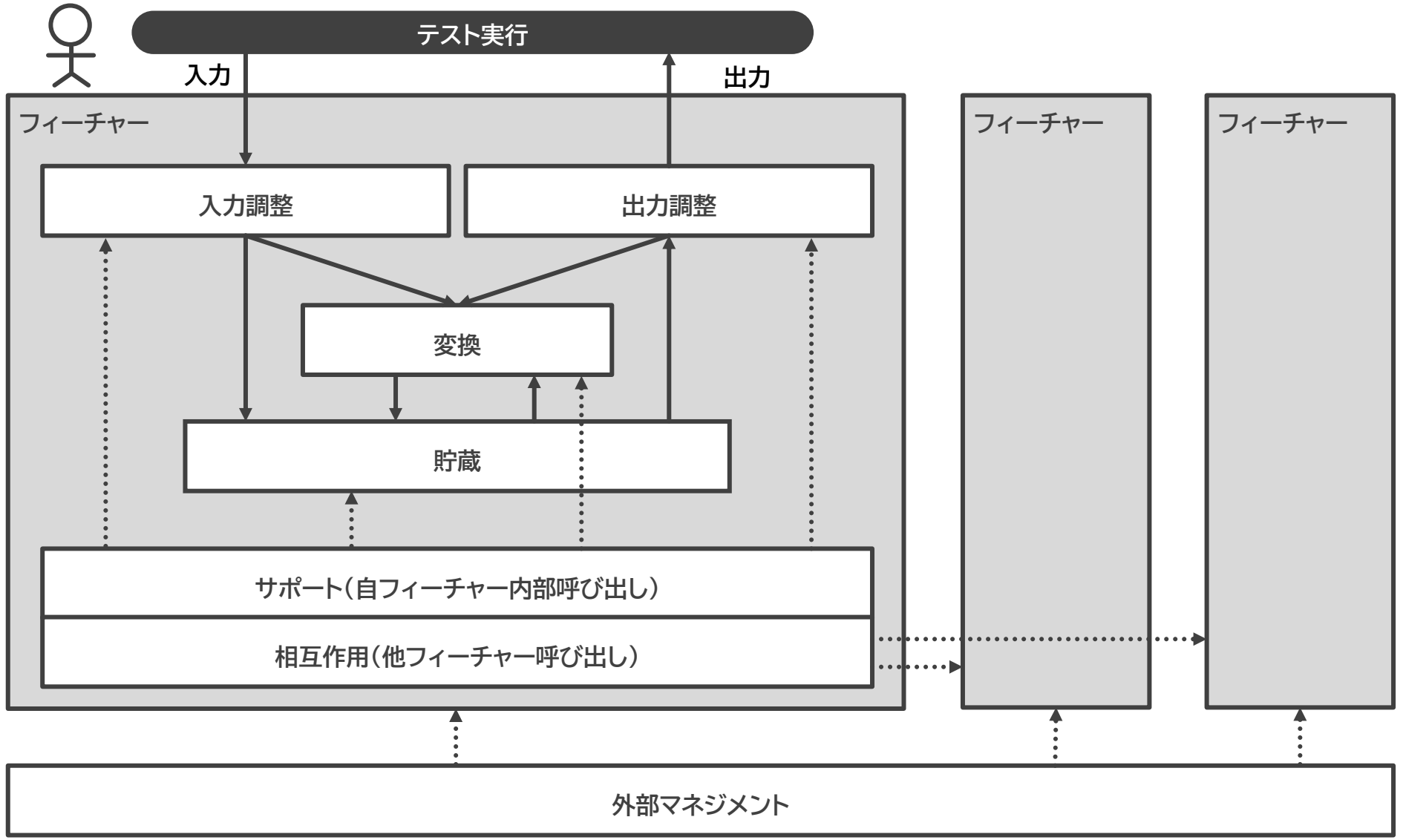
「(2021年版) ソフトウェアテストの上流設計-4 テスト分析と論理的機能構造」湯本剛氏)の図表をもとに、一部変更

外部観察できる仕様

内部に関わる仕様

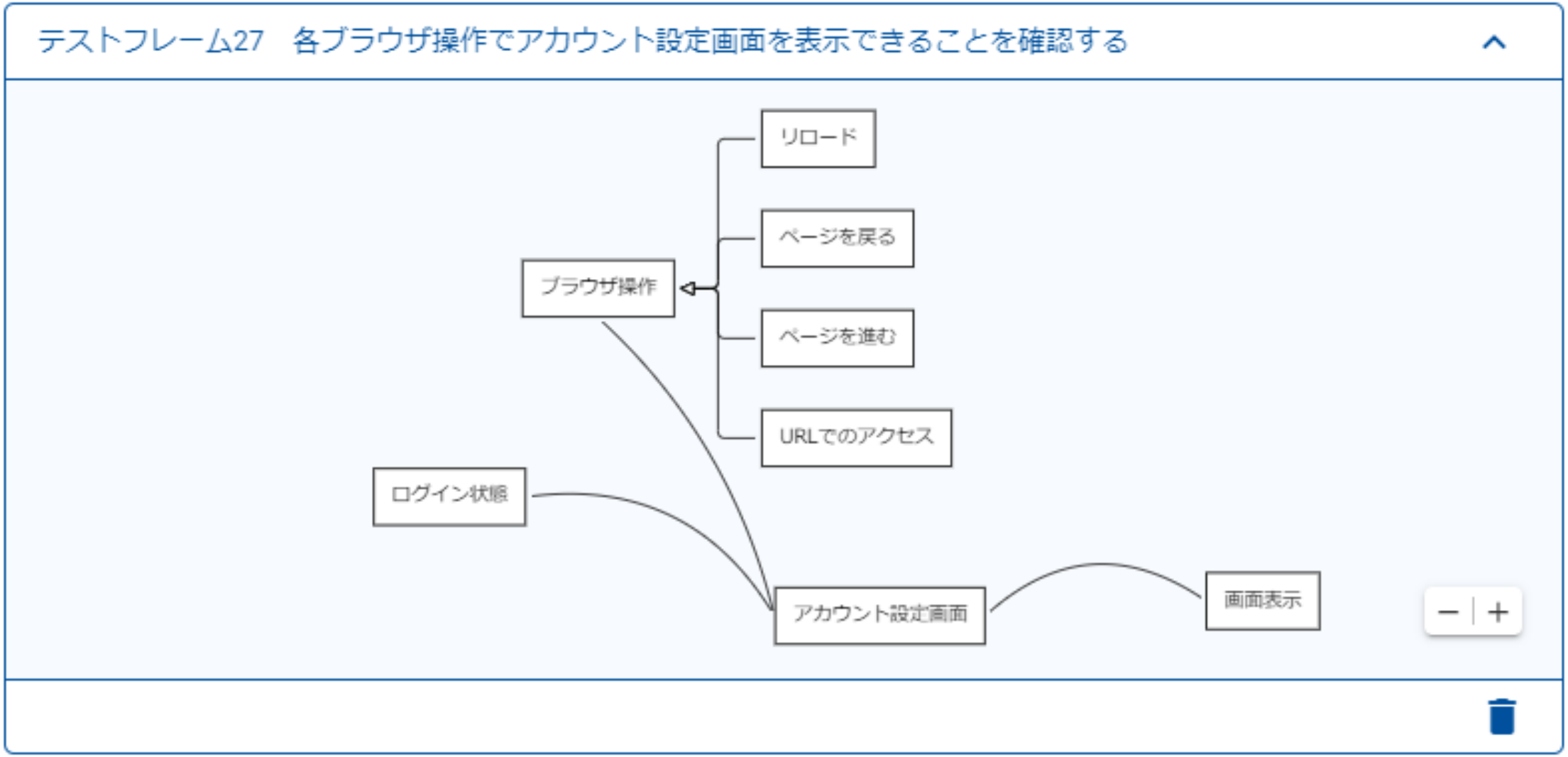
機能組合せ

AUT*外部



*AUT: Application Under Test

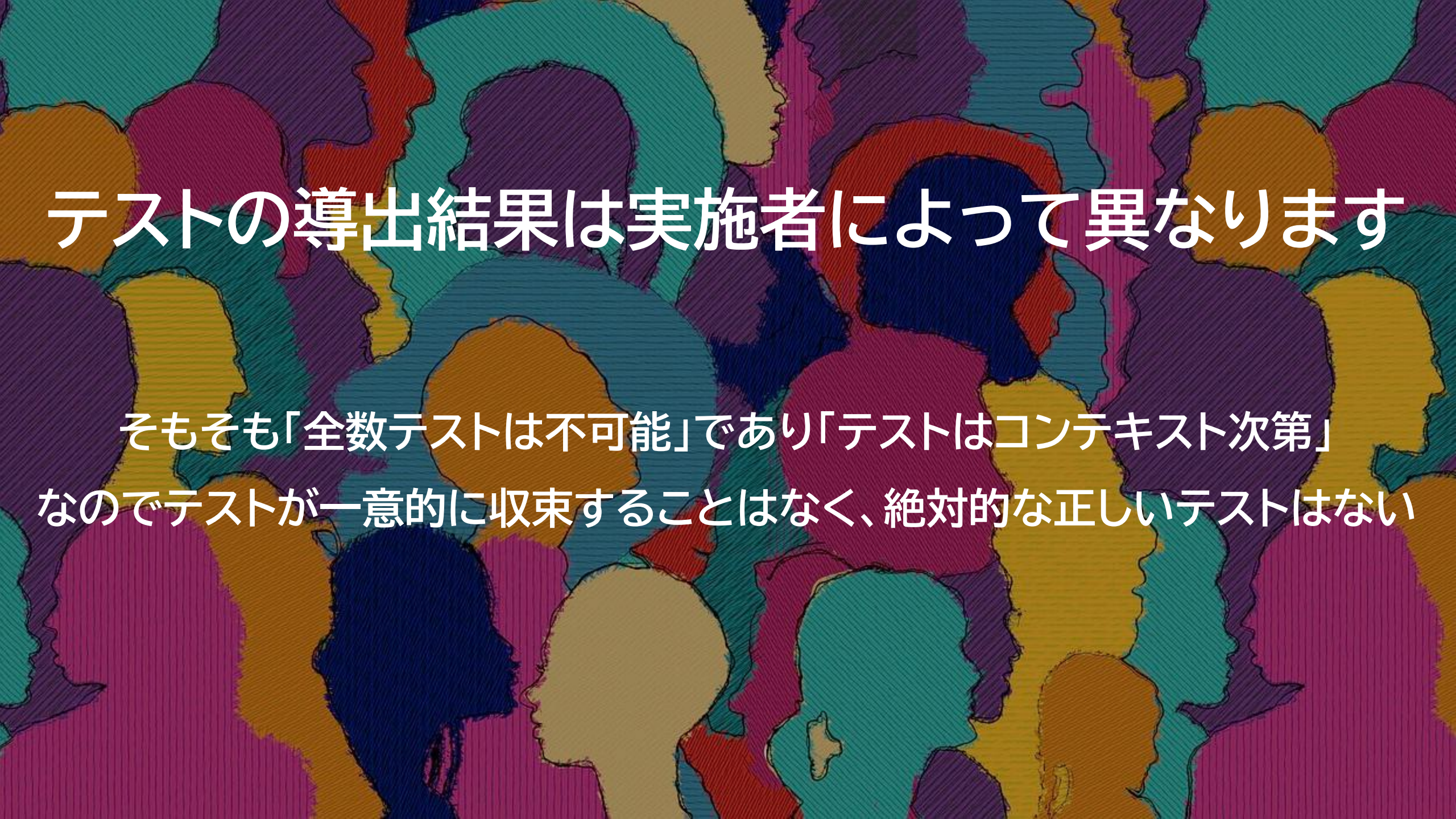
VSTePによるテストフレーム



GIHOZブログ「VSTeP適用事例のご紹介」より引用

なぜ、これらのテンプレート的なモデルを使うと便利なのか？

- テスト実行する単位に基づいて考えたいので
  - 動的テストの基本は、テストの対象に入力を与えて出力を観測することである
  - つまり、テストを実行するにあたり、入出力のインターフェースを識別するとよい
- それに影響を及ぼす要素を識別したいので
  - テスト対象によっても、どのように要素を構造化すると便利なのかは異なる
  - 将来的にはテスト対象の特性に合ったモデルを、自ら作れるようになるとよい



テストの導出結果は実施者によって異なります

そもそも「全数テストは不可能」であり「テストはコンテキスト次第」
なのでテストが一意的に収束することはなく、絶対的な正しいテストはない

どんなテストをするのかを利害関係者とすり合わせて合意を得ることが重要であり、そのためにはテストを説明できる必要がある

テストのスキルだけでは駄目で、バランスが大事！

テスト担当者のスキルスペース

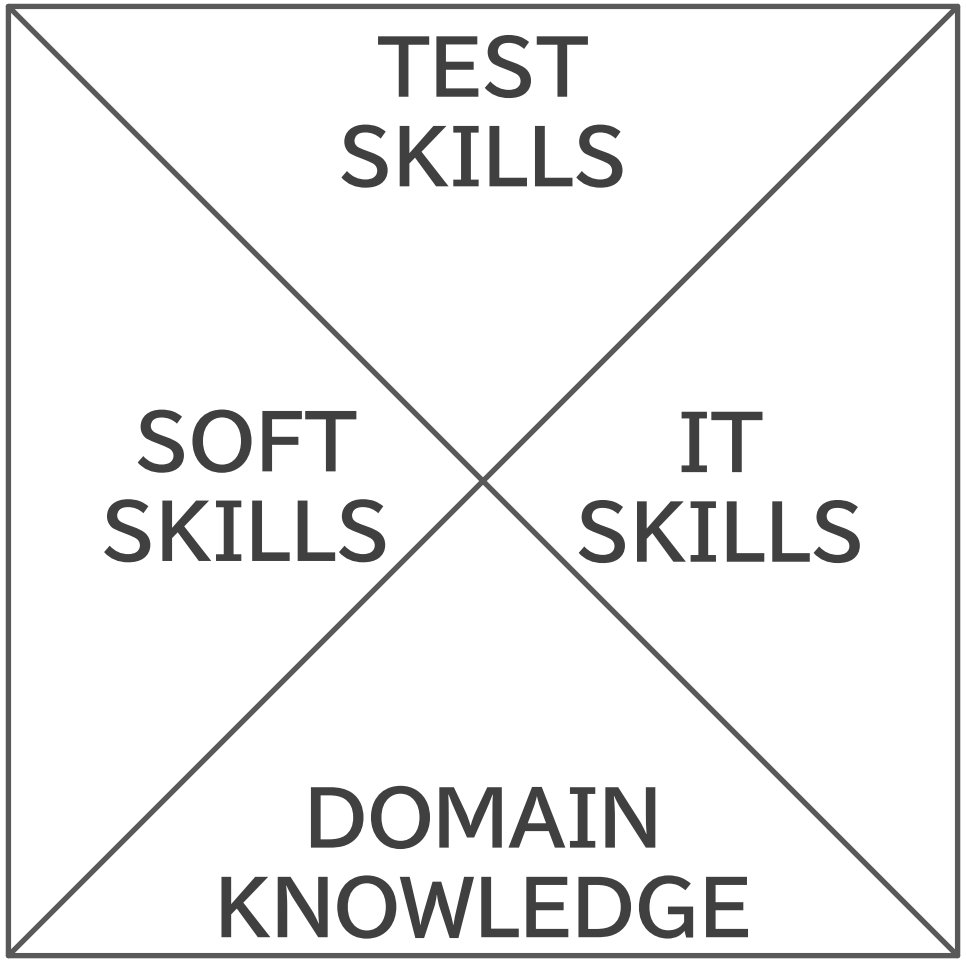


Fig. “Tester Skillspace” by Stuart Reid from JaSST Tokyo’14 Session Material (H8-2-2.pdf)

深く潜るためにもテスト担当者には
テスト以外のスキルや知識も重要です

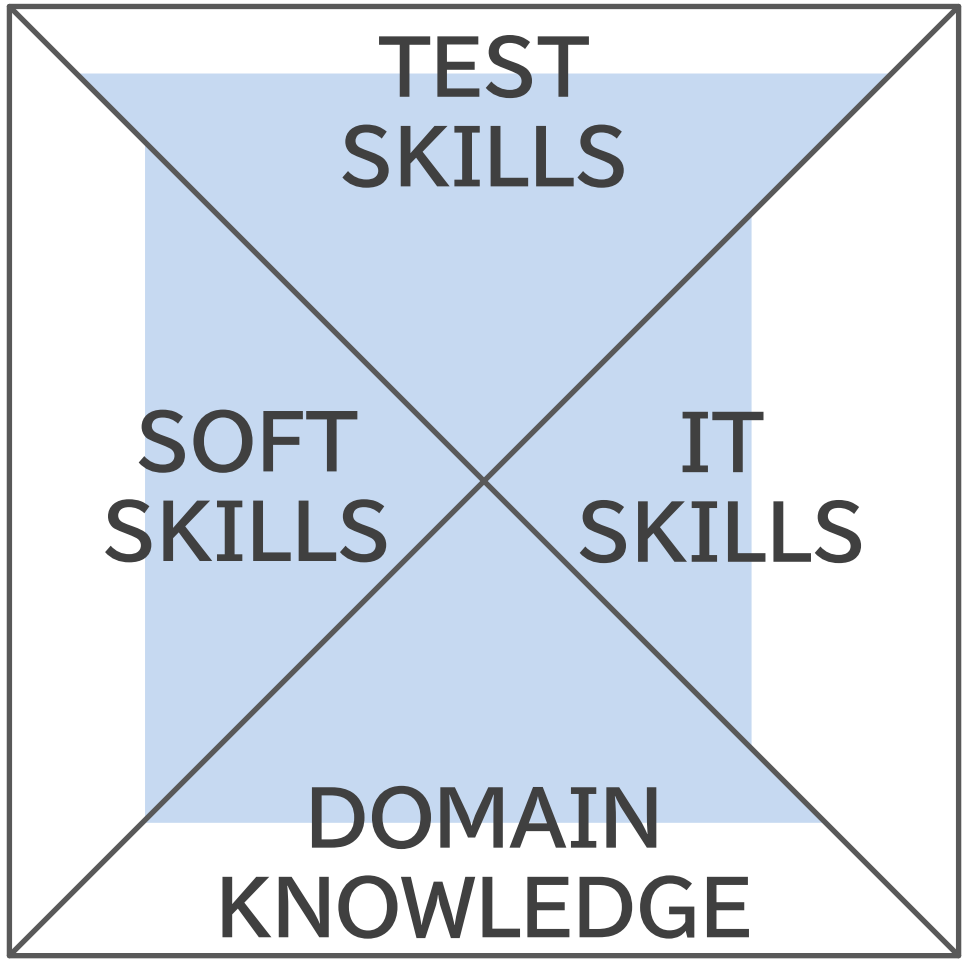
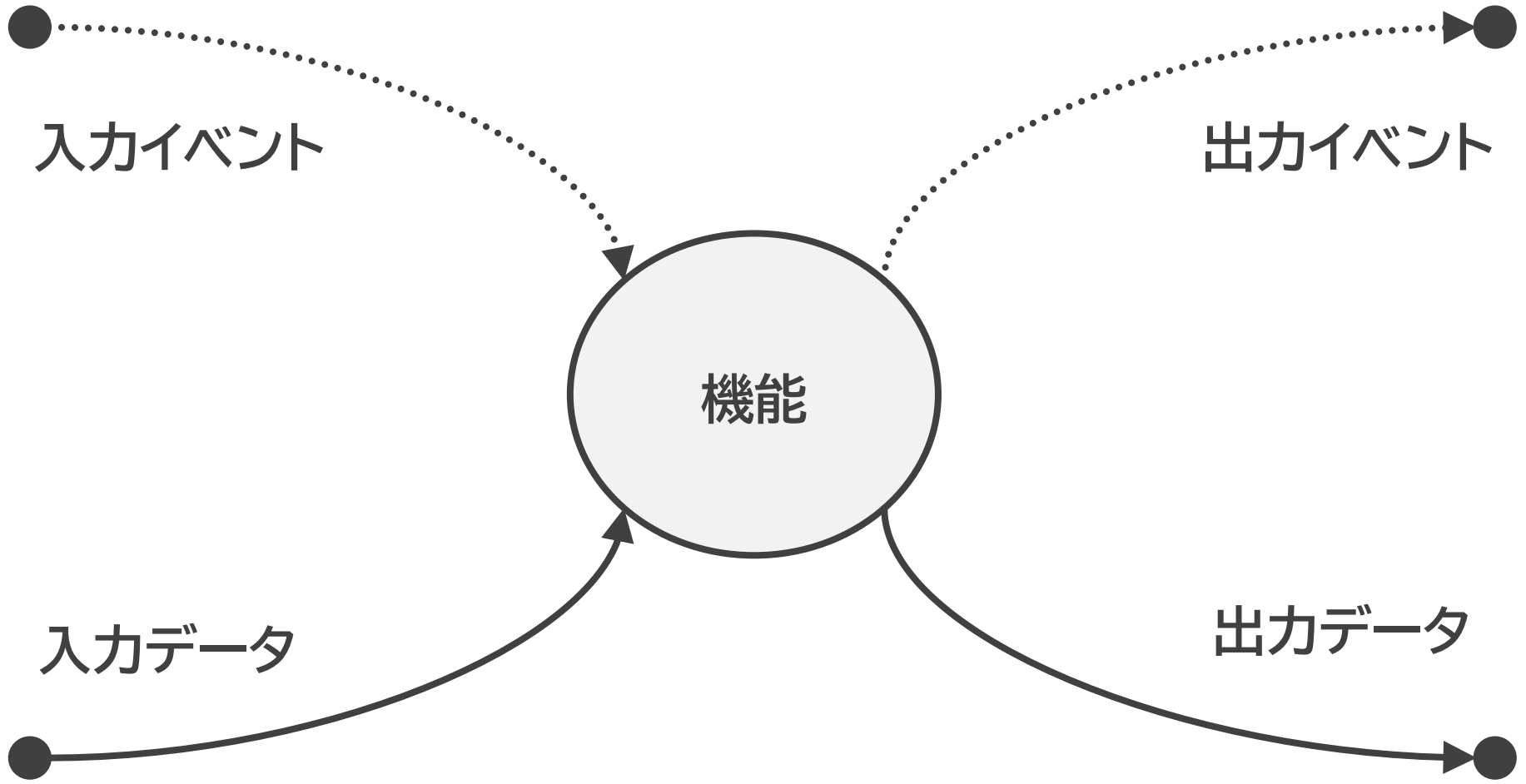
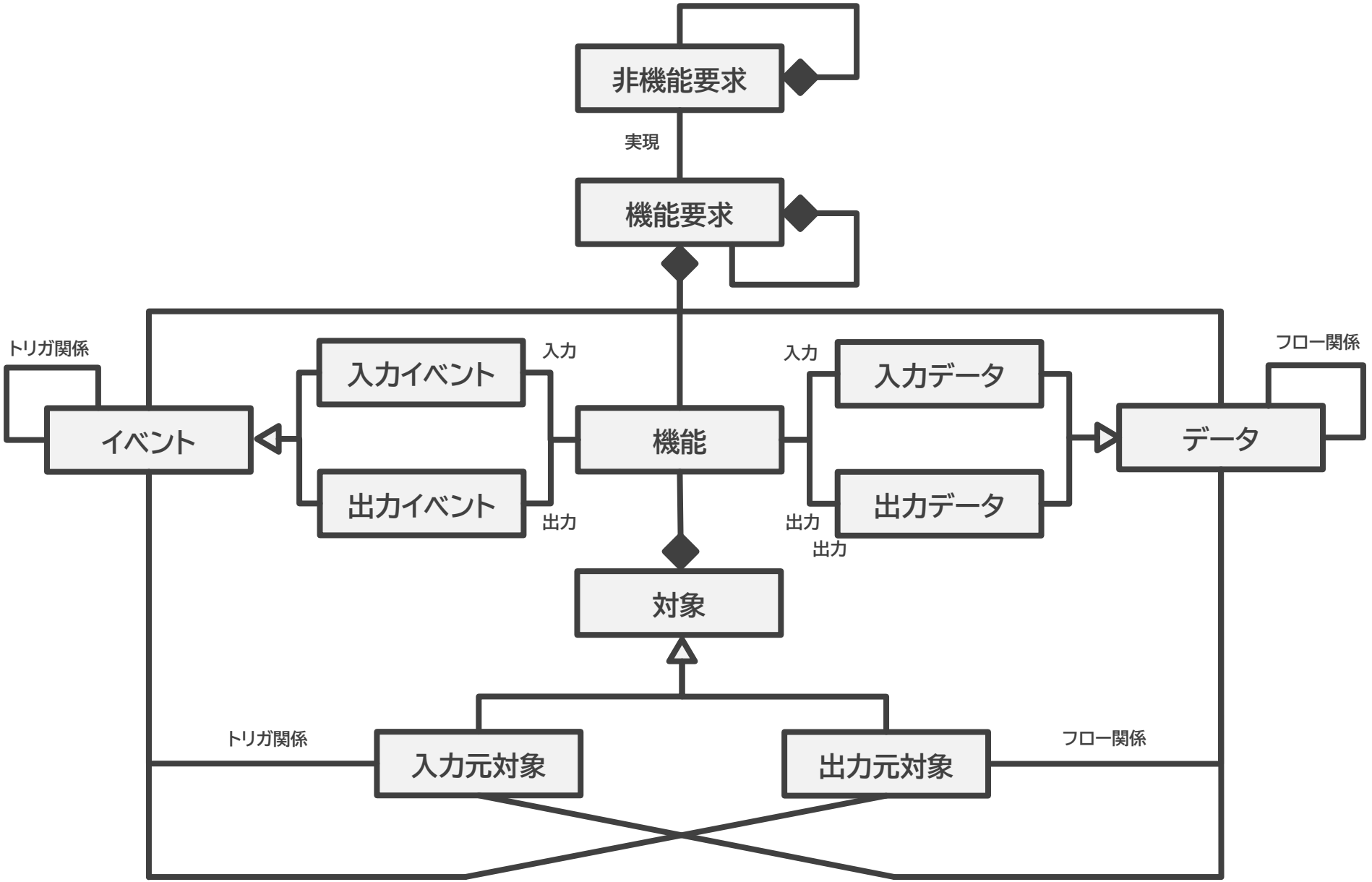


Fig. “Tester Skillspace” by Stuart Reid from JaSST Tokyo’14 Session Material (H8-2-2.pdf)

理想は、要求の段階から
整理されているとよいよね



P.77「図 5-2 一般的な機能に対するコントロールフロー図 」in「要求プロセスと技法入門要求開発の基礎知識」山本修一郎著（2019）



「要求プロセスと技法入門要求開発の基礎知識」山本修一郎著（2019）P.78「図5-3 要求のメタモデル」より、一部改変

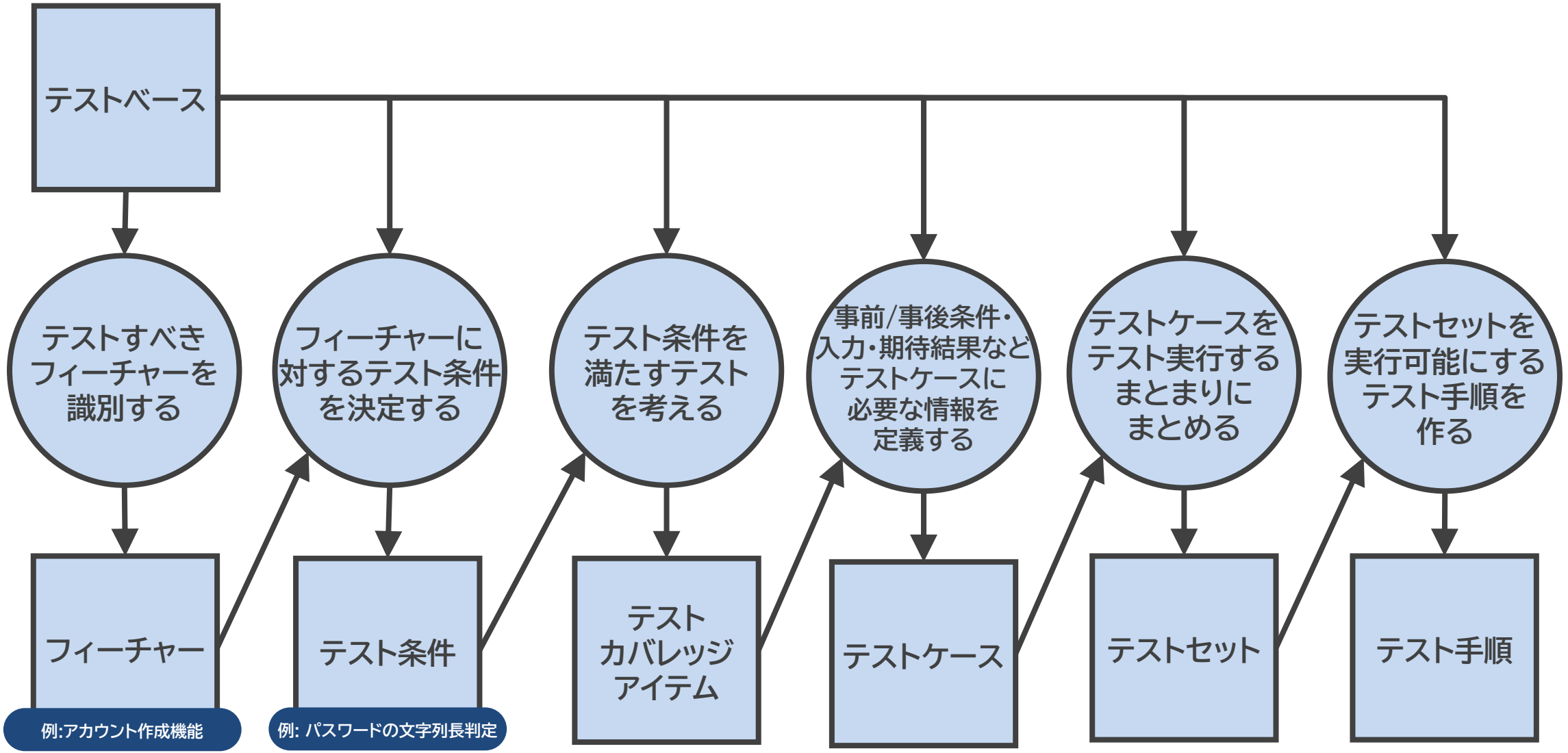
<table>
<tr><td>要求仕様</td><td colspan="4">危険運転を警告する</td></tr>
<tr><td colspan="2">契機</td><td>機能</td><td>応答</td><td>応答対象</td></tr>
<tr><td colspan="2">センサーログの更新</td><td rowspan="15">①センシングデータに基づき、危険運転を判別する。

• 速度超過運転
• ふらつき運転
• 急ブレーキ連続運転

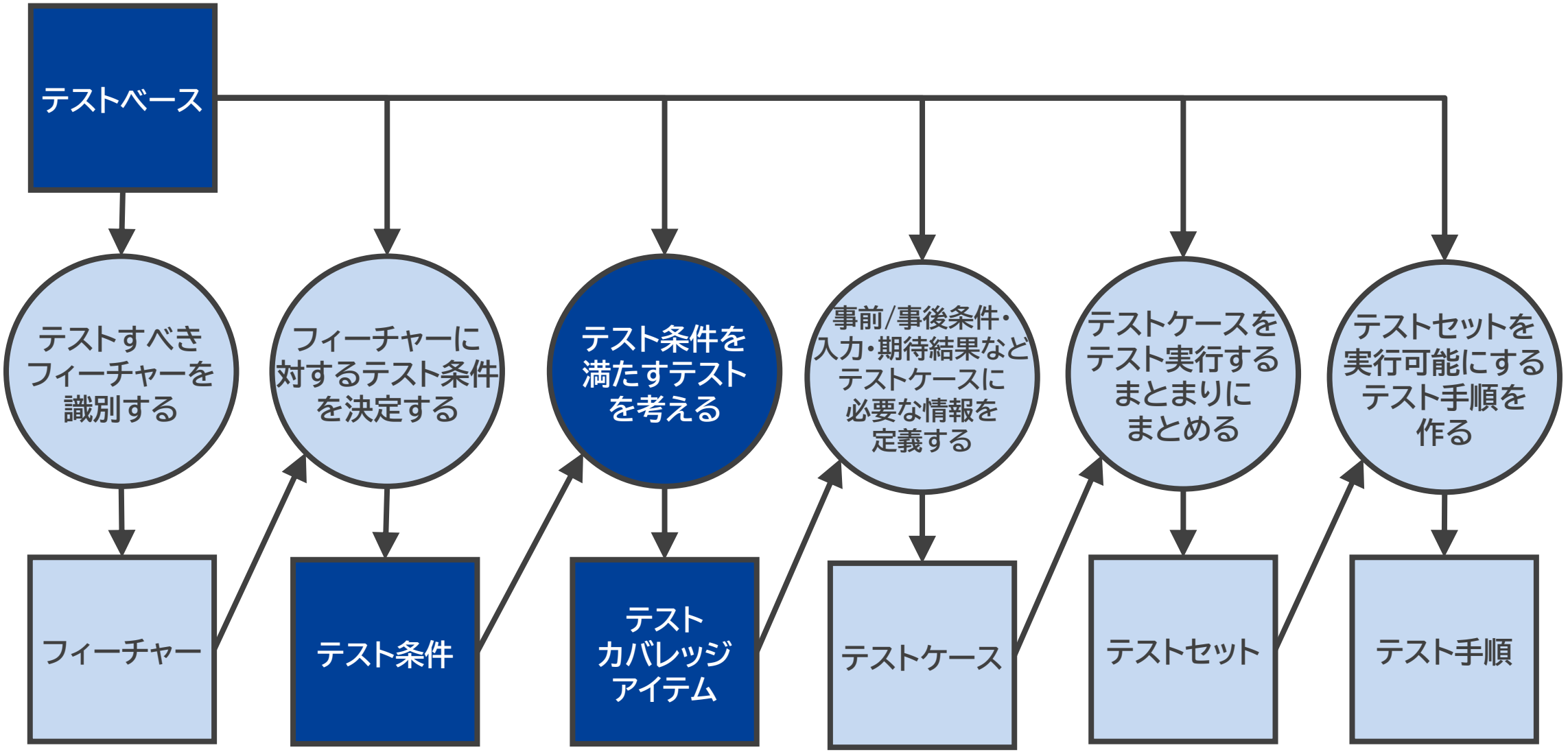
②運転者ならびに周辺者に指定された警告条件で警告する。</td><td>バイブレーション警告</td><td>ハンドル</td></tr>
<tr><td>入力対象</td><td>入力</td><td>音声警告</td><td>スピーカー</td></tr>
<tr><td rowspan="7">センサーログ</td><td>車輪の回転パルス</td><td>警告サイン</td><td>運転者用ランプ</td></tr>
<tr><td>ハンドルの回転角</td><td>警告サイン</td><td>簡易モニター</td></tr>
<tr><td>車体の傾き</td><td>周辺音声警告</td><td>周辺者用スピーカー</td></tr>
<tr><td>光量</td><td>周辺光警告</td><td>周辺者用ランプ</td></tr>
<tr><td>湿度</td><td>出力</td><td>出力対象</td></tr>
<tr><td>車体振動</td><td rowspan="8">危険運転警告記録</td><td rowspan="8">警告ログ</td></tr>
<tr><td>前方の物体</td></tr>
<tr><td rowspan="2">危険運転判断情報管理</td><td>自動走行条件</td></tr>
<tr><td>走行環境条件</td></tr>
<tr><td rowspan="4">危険運転警告条件情報管理</td><td>警告条件</td></tr>
<tr><td>-- タイミング</td></tr>
<tr><td>-- 間隔</td></tr>
<tr><td>-- 期間</td></tr>
</table>

「要求プロセスと技法入門要求開発の基礎知識」山本修一郎著（2019）P.80「表5-8 要求記述表の例」

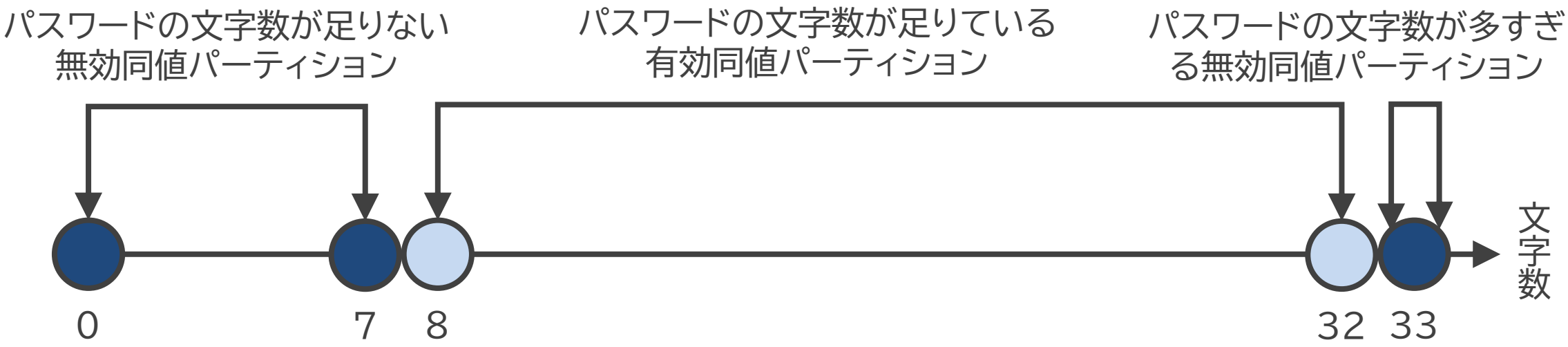
テスト開発プロセスをもう一段階深掘りしたPFD



テスト開発プロセスをもう一段階深掘りしたPFD



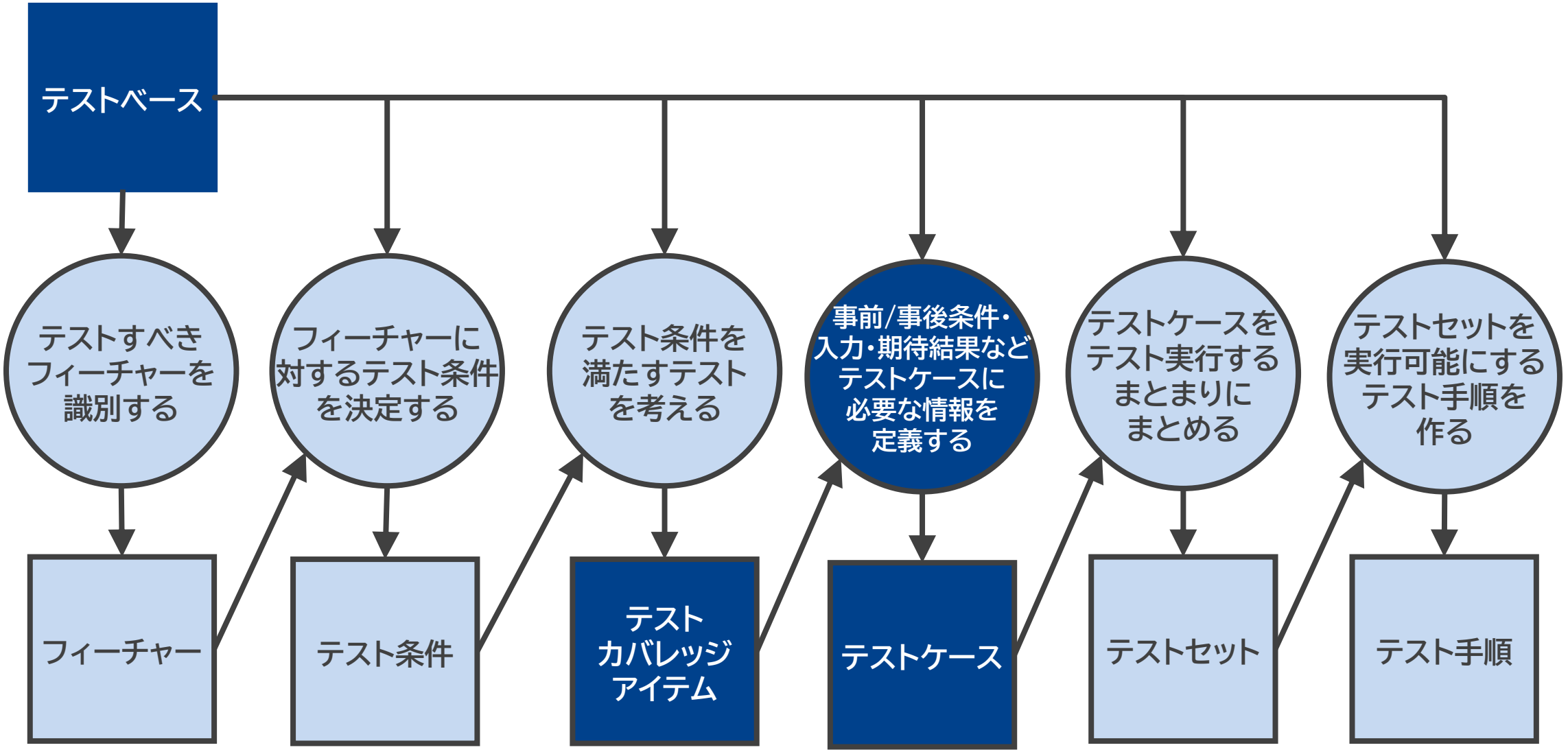
カバレッジアイテム(何を網羅すればテスト条件を確認したと言えるか考える)



1. パスワードの文字数が規定よりも足りない場合の最小値
2. パスワードの文字数が規定よりも足りない場合の最大値
3. パスワードの文字数が規定内の場合の最小値
4. パスワードの文字数が規定内の場合の最大値
5. パスワードの文字数が規定を超える場合の最小値

テストカバレッジアイテムを導出するために、多くの場合テスト技法を用いる。テスト技法については本稿では触れないが、多くの技法の導出結果はテストカバレッジアイテムである。

テスト開発プロセスをもう一段階深掘りしたPFD



テストケース（事前/事後条件・入力・期待結果を特定する）

#	境界値分析のカバレッジアイテム	事前条件	入力	事後条件	期待結果
1	パスワードの文字数が足りない場合の最小値	パスワード設定画面 未登録状態	0文字	パスワード設定画面 未登録状態	パスワード設定画面の「パスワード」テキストボックスの直下に、赤文字で「パスワードは8文字以上32文字以内の長さを指定してください」と表示され、設定完了画面に遷移しない
2	パスワードの文字数が足りない場合の最大値		7文字	パスワード設定画面 未登録状態	パスワード設定画面の「パスワード」テキストボックスの直下に、赤文字で「パスワードは8文字以上32文字以内の長さを指定してください」と表示され、設定完了画面に遷移しない
3	パスワードの文字数が足りているの場合の最小値		8文字	設定完了画面 登録済み状態	アカウントが作成され、設定完了画面へ遷移する
4	パスワードの文字数が足りている場合の最大値		32文字	設定完了画面 登録済み状態	アカウントが作成され、設定完了画面へ遷移する
5	パスワードの文字数が多すぎる場合の最小値		33文字	パスワード設定画面 未登録状態	33文字目のユーザーの入力が無視される ※テキストボックスに33文字目を入力できない

テストやテストケースという言葉の意味も曖昧な場合がある

<<<<<< 広義のテストケース >>>>>>

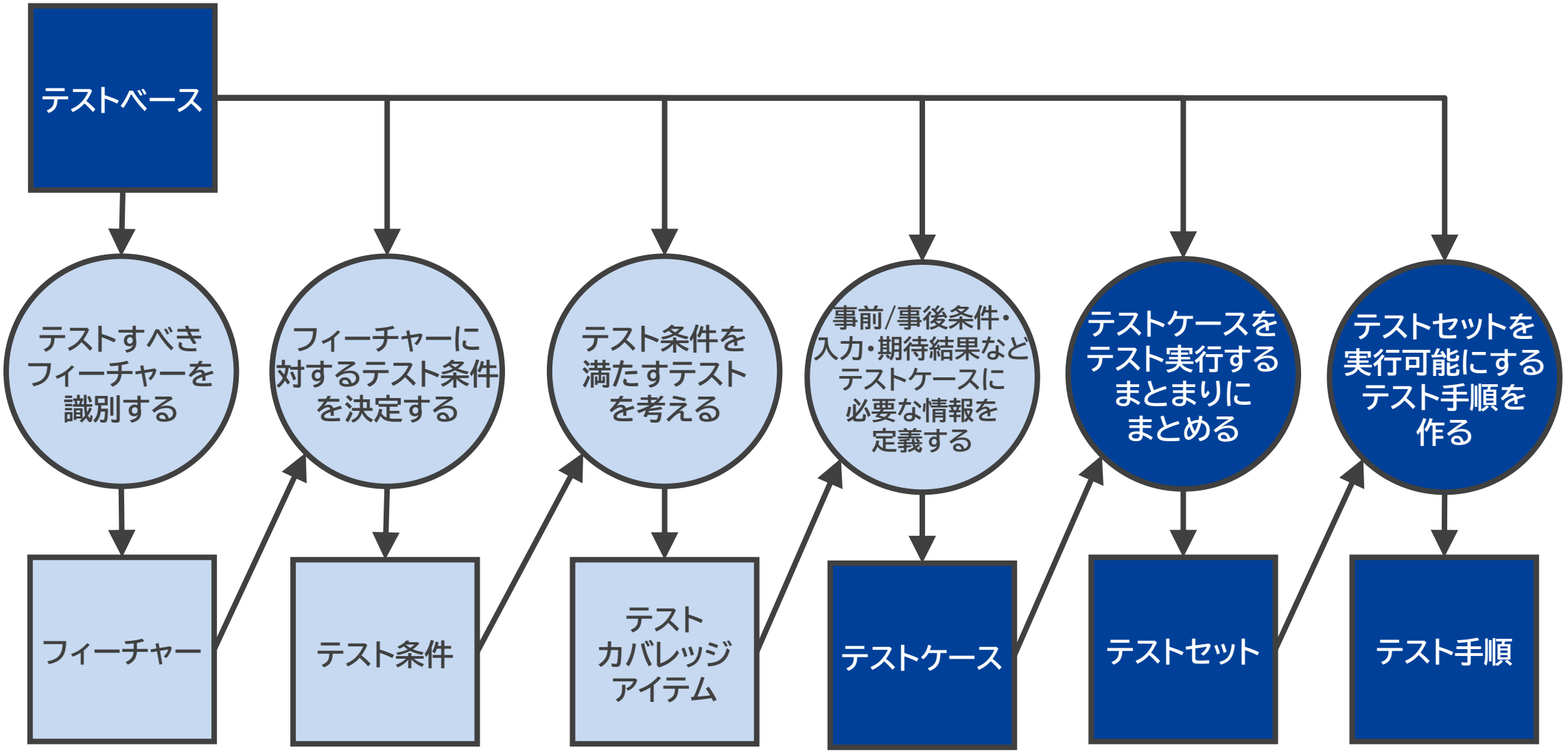
- テスト開発プロセスにおける成果物を丸っとテストケースと呼ぶ場合がある
  - テスト条件、テストケース、テスト手順などを明確に区別しないような文脈
- 実施するテスト内容を漠然とテストケースと呼ぶ場合がある
  - テスト(test)と同義語くらいの感じで、特段の意図なくテストケースと呼ぶ

>>>>> 狭義のテストケース <<<<<

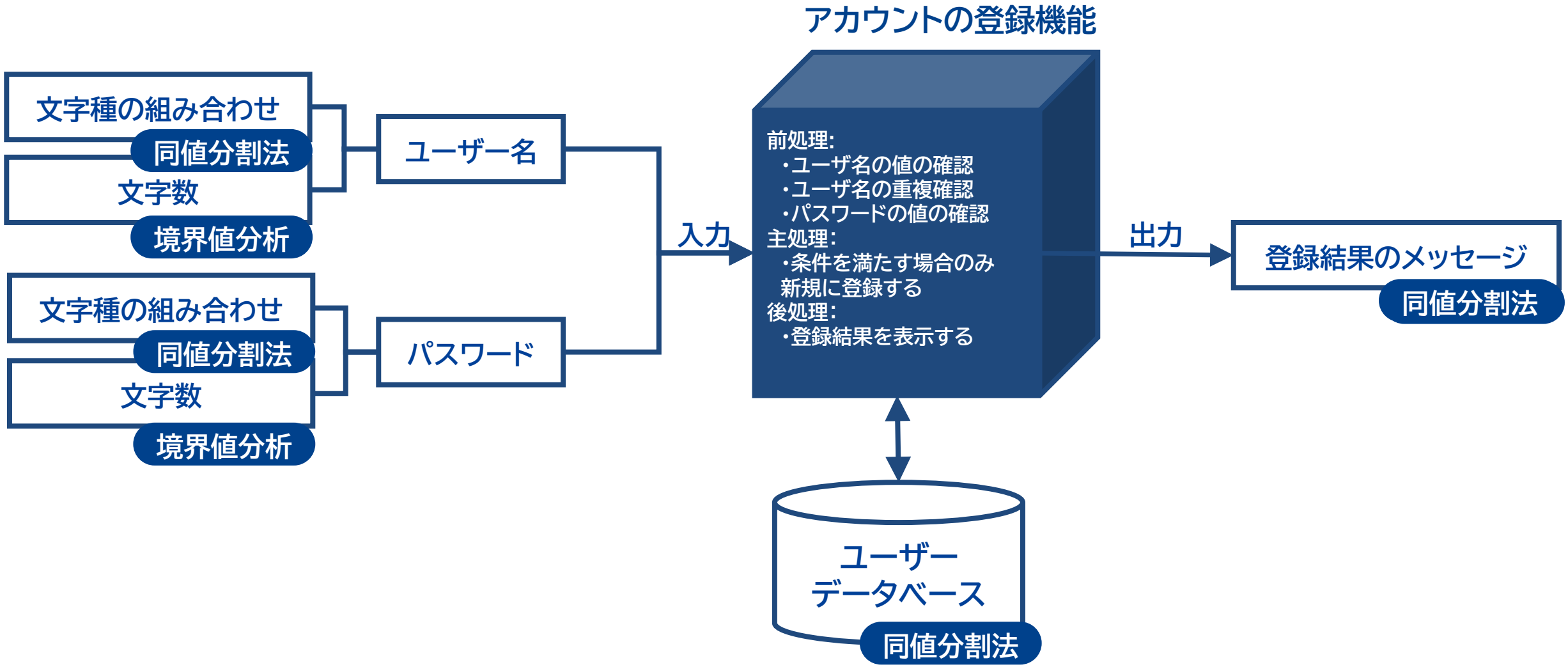
- テスト設計のアウトプット(成果物)
  - テスト条件、テストケース、テスト手順を明確に区別するような文脈
- 厳密に、事前/事後条件、入力、期待結果といった情報を含んでいるものを指す

ここで得られる示唆は、相手の言っていることと自分の言っていることが同じ用語でも異なる意味を持っている場合があるということ。また、そこで誤解を起こさないためにも用語の定義というのは地味なようでも非常に重要であるということ

テスト開発プロセスをもう一段階深掘りしたPFD



1回のテスト実行で、複数のテストケースを確認しうる



複数のテストケースを組み合わせることのメリットとデメリット

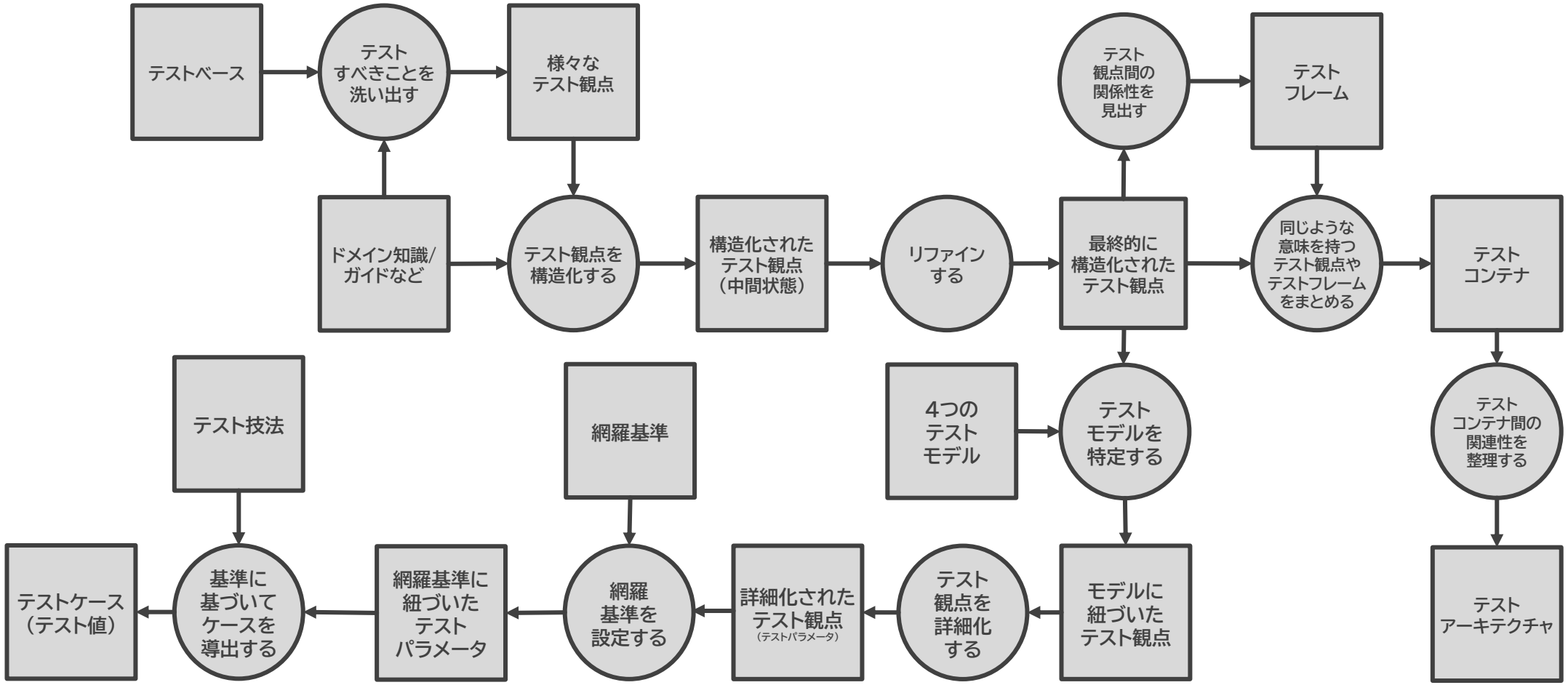
メリット

1つのテスト手順を**実行するコストが高い場合、少ない実行回数によってカバレッジ**できる。
例えば、業務システムにおけるシステムテストでは、1つのテスト手順を実行するために、数時間や数日を必要とする場合がある。

デメリット

仮にそのテストセットを実行したことによって故障を識別した場合、**問題の切り分け**（どのテストケースが原因で、テスト手順が失敗したのか）が、**難しくなったり手間取ったり**する。
そのテストセットがどういった意味のテストであるのか、**テストの意味合いがぼやける。**

どうやれば自分達のコンテキストでうまくテストを作れるか、テスト開発プロセスを改めて再整理してみましょう



生成AIの性能向上が著しい昨今、作業的なものはどんどん生成AIが担うことになります。
その中で人に求められるのは、生成AIへ与える指示と、その結果の妥当性を判断することです。
また、結果の妥当性を判断するためにも、AIに一足飛びで出力させるのではなく、段階的に中間成果物を出力させ、適宜確認し、修正します。
そして、どのような段階を踏ませてどのような中間成果物をどのように作らせるかは、人が示す必要があります。
つまり、テスト開発のプロセス設計ができないと、AIの作った成果物について妥当かどうかを判断しがたくなります。

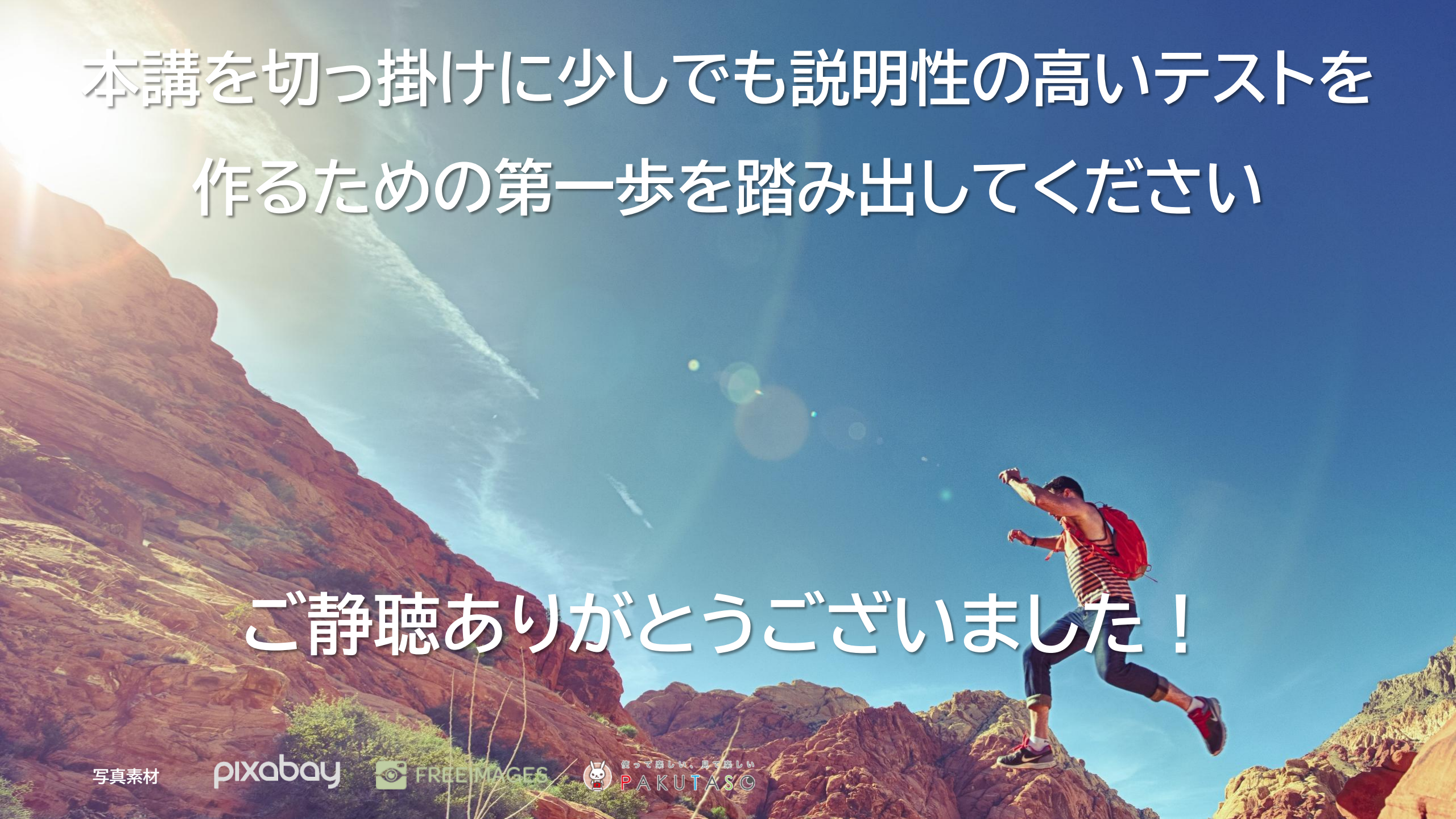
この講演のゴール

テスト設計について改めて考えてみて、

テストベースから説明可能なテストケースを

導く考え方の足掛かりを得る

本講を切っ掛けに少しでも説明性の高いテストを作るための第一歩を踏み出してください



ご静聴ありがとうございました！

写真素材 pixabay FREE IMAGES 使って楽しい、見て楽しい PAKUTASO

主な参考文献一覧

文献	著者	発行年
ISO/IEC/IEEE 29119-2:2013 Software testing Part 2: Test processes	ISO/IEC/IEEE	2013
ISO/IEC/IEEE 29119-2:2021 Software testing Part 2: Test processes	ISO/IEC/IEEEE	2023
テスト技術者資格制度 Foundation Level シラバス Version 2023V4.0.J02	ISTQB	2024
テスト設計チュートリアル 2021 ちびこん編資料	山﨑崇, 西康晴	2021
WACATE 2023 夏 ー テスト開発について改めて考えてみよう！ ～ テスト分析やテスト設計を業務で上手くできるようになるための四方山話～	山﨑崇	2023
JaSST’10 東京 テスト初心者セッション～テス活はじめてみよう～	長谷川聡, 片山徹郎	2010